the i-engineers

Customizing

i-engine

V8.7.09 2011-06

Personenbezeichnungen gelten gleichermassen für Personen beider Geschlechter.

Disclaimer

Das Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere der Übersetzung, des Nachdrucks, der Entnahme von Abbildungen, der Weitergabe auf photomechanischem oder ähnlichem Weg und der Speicherung in Datenverarbeitungsanlagen bleiben, auch bei nur auszugsweiser Verwertung, sind vorbehalten.

Trademarks: Windows ist eine Trademark von Microsoft Corporation. Andere Markennamen und Produktenamen sind durch die jeweiligen Inhaber geschützt

Die in diesem Handbuch beschriebene Software wird unter einem Lizenzabkommen geliefert und kann nur unter den Bedingungen dieser Lizenz verwendet werden. Die Informationen in diesem Handbuch können jederzeit geändert werden. Sie stellen keine Verpflichtung in Bezug auf die Funktionalität von Seiten *the i-engineers AG* dar. *the i-engineers AG* kann jederzeit Erweiterungen und/oder Änderungen an diesem Handbuch oder der *i-engine* Software vornehmen.

Copyright Hinweis:
© 2011 the i-engineers AG
Rain 42
CH-5000 Aarau; Switzerland

Alle Rechte vorbehalten

Hergestellt in der Schweiz, produziert von bbi 2011

Inhaltsverzeichnis

Generell **9**

Inhalt dieses Buches 9

Typographie 10

Customizing **11**

Vorgehen 11

Wie wird i-engine parametrisiert 12

Ordnerstruktur 13

Basismodul 13

Module 18

Kunde 19

Datenablage 21

Wie kann die Leistung verbessert werden? 21

Generelle Arbeiten **22**

Standard Operationen 22

Datenbank Standards 24

Nomenklatur Oracle Bezeichner 29

SQL-Statements 33

Erzeugen und Speichern 34

Nomenklatur SQL-Statements 36

SQL-Statement Typen 36

Variablen 40

Before-Statement 40

Expiration-Statement 44

Fork-Statement 44

Parameter-Statement 48

Finder & Constraint-Statement 52

Default-Statement 53

Value-Statement 53

Dyn-Statement 54

PL/SQL-Funktionen 55

Nomenklatur PL/SQL-Funktionen 55

Speichern und Verwenden 56

Werteliste erzeugen 58

Logging 60
Geschäftsprozesse 62
Der Prozess 62
Einen neuen Prozess erstellen 63
Der Prozessstatus 65
Einen neuen Status erzeugen 65
Katalog - Standard Symbole für Prozessstatus 70
Statusbezeichnungen 70
Zwischenstatus 72
Die Aktivität 74
Genereller Ablauf einer Aktivität 75
Eigenschaften 77
Nomenklatur 89
Rollen zuteilen 96
Aktivitätsgruppen festlegen 97
Die Methode 99
Ereignisse - Event Handling 102
Unified Messaging System - UMS 105
Prinzipieller Ablauf 106
Implementierung UMS 107
Eintrag in IPC Tabelle 108
Eintrag in Subscription Tabelle 109
Eintrag in Service Tabelle 111
Auslösen der Meldung 112
E-Mail Versand (veraltet) 113
E-Mails im Workflow 113
E-Mail Vorlage 114
Workflow Verknüpfung 115
E-Mails Abonnieren 117
Versenden nach Ablauf 118
Hinweise 119
E-Mails in Überwachung 120
Parametrisieren E-Mails in Überwachung 121
Datenwächter 122

Eingabemasken und Listenanzeige **124**
Maskenaufbau 125
Nomenklatur 126
Standard Attributprofile 127
Attribute bearbeiten 128
Hinweise zu Show-Typen 133
BUTTON – Formular Schaltflächen 133
TAB – Registerblätter 135
HDR-File – Daten für Drittanwendungen 137
HIDDEN – verborgene Felder 137
CHOSE / CHOOSEN – mehrere Auswahlfelder 138
Maske für neue Objekte 140
Tipps & Tricks 142
Eingabe Validierung 145
Daten-Checker 145
Server-Checker 147
Menüleiste & Strukturanzeige **149**
Konfigurieren der Menüleiste 150
Startobjekt 150
Aufzurufende URL 151
Temporärverzeichnis 152
Was wird im Baum angezeigt? 152
Objekte im Datenfenster anzeigen 153
Strukturanzeige modifizieren 155
Objekte – Dokumente & Ordner **157**
Objektklassen abgeleitet aus Objektklassentypen 158
Aufstellung Objektklassentypen 160
Eine Objektklasse erzeugen 164
Zugriffsrechte auf neue Objekte erteilen 168
Bildsymbole für die Objektklasse 168
Katalog - Standard Symbole für Objektklassen 169
Symbol Aktionen 170
Objekt Extension 172
Variablen für MS-Office Anwendungen 177
Ablage-, lokale Erweiterung, MIME-Typ 179

Einstellung der Speicheroption und Dateisäuberung 179
Vorlagedateien 180
Vorlagen für Objekte 182
Vorlagen-Objekt 182
Verwalten mehrerer Vorlagentypen in einer Objektklasse 184
Versionsverwaltung 186
Wozu sind Link/Alias Objekte? 187
Alias Objekte 188
Rollen und Benutzer 189
Rollen erzeugen 189
Benutzer erzeugen 192
Die Benutzervorlage 194
Passwort Einstellungen 194
Fenster 196
Hauptfenster 196
Projektfenster 197
Festerframes 200
Eingabeelemente 201
Textanpassungen 203
Integration 204
Single Sign-On 204
Authentifikation mit iTAN 207
Exchange/Outlook Anbindung 209
WebDav Anbindung 211
Konfigurieren OracleDrive 213
Konfigurieren Windows WebFolder 214
Datenimport 216
Verknüpfte Tabellen 216
Importprozedur 217
HL7 Schnittstelle 217
DICOM Schnittstelle 219
Integration in HTML-Seite 221
Direkter URL Aufruf 221
KIS-Link - Integration in Klinik Informationssystem 223
Vergleich 225

WebServices 226
Docbox Service erzeugen 226
Messages definieren 227
Datenaustausch initiieren 228
Forms Handling 229
InfoPath 229
Spezielle Konfigurationsaufgaben 232
AutoUpdate 232
Individualisierung 236
Zeitplanen von Aufgaben 237
Die Ablage 238
Organisation der Datenablage 238
Eine neue Dateiablage 239
IPC - Interprozess Kommunikation 243
Synchron/Asynchrone Aktivitäten 247
Sicherheitsklassifikation 249
Benachrichtigung Konfigurieren 252
Projekt Methoden 254
Projekt Methoden anwenden 254
IPC und Projekt-Methoden kombinieren 258
Konfigurationsbeispiele 259
Cut&Paste Operationen 259
Objekt Transfer 259
Fixer Prozessstatus 260
Prozesswechsel 262
Projekt 266
Dynamischer Prozesswechsel 267
Distribution 268
Dynamische Systemrollen 270
Aufsetzen der Parameter 271
Unterschied Distribution – SubDistribution 274
Clone Methoden 275
Sucher 277
Volltextsucher 281
Constraint - Filter 282

Reports 284

Pop-up Fenster definieren 288

Resultatfeld über Auswahlliste füllen 291

Kollektoren 297

Verwendungsnachweis 298

Berechtigte anzeigen 300

Häufig gestellte Fragen 302

Anhang 307

Merkblatt Nomenklatur 307

Applikations- und wichtige System-Objektklassen 309

Tabellen in i-engine 314

Applikation Symbole für Kundenprojekte 316

Glossar 323

Index 331

Generell

Inhalt dieses Buches

Dieses Handbuch richtet sich an die Parametriker der *i-engine*. Es enthält Anleitungen und Informationen zur Parametrisierung des Systems auf unterschiedliche Anforderungen. Für diese Aufgabe sind vertiefte Detailkenntnisse zu allen Aspekten von *i-engine* notwendig. In diesem Kapitel Handbuch sind die Basisinformation enthalten, die der Parametriker für das Customizing benötigt.

Weitere Informationen finden Sie in folgenden Unterlagen

- Systemkonzept - Leitgedanken und Grundkonzepte der *i-engine*
- Realisierungsbeispiel - Schritt-für-Schritt Anleitung zur Erstellung eines Beispielprozesses
- Systemdesign - die auf *i-engine* aufbauenden Lösungen wie he-engine, pe-engine oder be-smart sind in den jeiligen Systemdesigns dokumentiert.
- Benutzer-Kurzdokumentation - Schnellübersicht für Anwender des Flex Clients

Typographie

Element	Darstellung
Menüeintrag im Text	**Schaltfläche**
Aktivität im Text	**Editieren**
Prozessstatus im Text	«aktiv»
DB-Tabellen	SQL_STMT
DB Tabellen Felder (Text/Auflistung)	OBJECT_ID
Maskenfeld im Text	*Beschreibung*
Eingabewert (Text/Auflistung)	«YES»
Kapitelreferenz	s. Abschnitt «Konfiguration»
SQL-Statements	`SELECT * FROM DUAL`
Ordner	«Dokumente» «Kunde > Parametrisierung > Prozesse»
Strings	«Hallo» «@» Zeichen oder Textstrings werden von Doppelklammern eingerahmt.
Begriffe	«Werteliste»
Methode	*Object.Edit()*

Customizing

Beim Bau kundenspezifische Prozesse mit *i-engine*, zeigt sich die grosse Flexibilität und Stärke des Systems.

- Das System kann durch einfaches Parametrisieren in einer grossen Bandbreite den Prozess- und Dokumentationsanforderungen angepasst werden.
- Auch für das Parametrisieren genügt der einfache Zugriff über den Web-Browser. Dabei wird keine neue Anwendung oder unterschiedliche Bedieneroberfläche verwendet, sondern alle Arbeiten werden in der gewohnten *i-engine* Umgebung ausgeführt. Die einzige Voraussetzung ist, dass der Benutzer die notwendigen Berechtigungen besitzt.
- Viele Änderungen können im laufenden Betrieb durchgeführt werden. Neue Prozesse oder geänderte Abläufe werden sofort wirksam. Betriebsunterbrüche, Neustarts etc. sind nicht mehr nötig.
- Auch die Module und Basisobjekte selbst sind als Objekte in *i-engine* abgelegt. Sie können von ausgebildeten Konfigurationsexperten ebenfalls modifiziert werden.

Vorgehen

Was kann durch Customizing angepasst werden?

Customizing der Module

Jeder Applikation wie *public-engine* oder *health-engine* umfasst eine Reihe Module, welche in standardisierter Form ausgeliefert werden. Der Funktionsumfang, die detaillierte Spezifikationen des Datenlayouts oder Arbeitsablaufes und der Maskenaufbau sind im jeweiligen Systemdesign festgehalten. Im Rahmen des Customizings können diese Grund- und Applikationsmodule auf die individuellen Bedürfnisse des Kunden zugeschnitten werden.

Customizing des Layouts

Während die prinzipielle Bildschirmdarstellung durch die Module vorgeben ist, können grafische Parameter wie Farbe, Schriftart oder Grösse weitgehend angepasst und die Anwendung damit dem Erscheinungsbild des Unternehmens angeglichen werden. Sämtliche verwendeten Texte können mehrsprachig verändert und erweitert werden.

Integration in Kundenumgebung

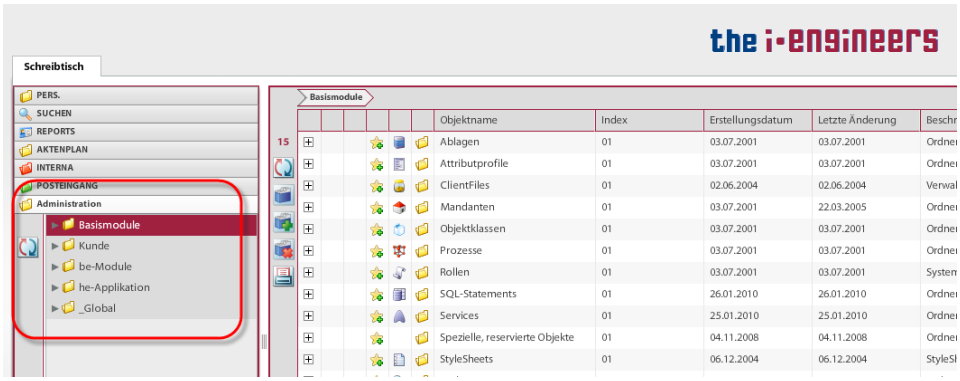
Die Integration der Applikation in die vorhandenen Systeme des Kunden ist entscheidend für die Akzeptanz und die Einfachheit der Verwendung. Die Verknüpfung mit Verzeichnisdiensten und Office Anwendung ist deshalb ein häufiger Customizing Schritt.

Was wird beim Customizing gemacht?

i-engine ist eine Datenbank basierende Lösung. Customizing bedeutet, dass Inhalte von Parameter-Tabellen und SQL & PL/SQL-Befehlen oder Tabellenstrukturen geändert oder erstellt werden. Meist können diese Änderungen bequem über spezielle Webseiten der *i-engine* vorgenommen werden. In einigen Fällen ist aber ein direkter Zugriff auf die zugrunde liegende Datenbank notwendig.

Wie wird i-engine parametrisiert

Benutzer mit der entsprechenden Berechtigung erhalten eine erweiterte Menüleiste, welche den Zugriff auf die Parameter der *i-engine* ermöglicht. Über den Menüpunkt **Administration** öffnen Sie die Anzeige der Parameterdaten im Strukturfenster. Damit haben Sie Zugriff auf alle Objekte, welche beim Customizing bearbeitet werden.

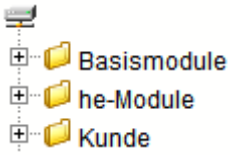


So verwenden Sie *i-engine* bei der Parametrisierung:

1. Melden Sie sich am System mit einem Benutzer an, der die notwendige Berechtigung für das Customizing besitzt. Alternativ können Sie auch als Stellvertreter eines Benutzers mit entsprechender Berechtigung agieren.
2. Verwenden Sie den Menüpunkt **Administration** im Strukturfenster. Diese Verknüpfung zeigt auf den Parametrisierungsteil der Applikation, für welchen der angemeldete Benutzer verantwortlich ist.
3. Im Strukturfenster können Sie nun zu den betreffenden Einstellungen navigieren.

Ordnerstruktur

Alle Objekte für das Customizing sind im Parameterbaum abgelegt. Die vom Benutzer einseh- und bearbeitbaren Elemente hängen von seiner Berechtigungsstufe ab.



Basismodul: Alle Basisobjekte, Prozess, Rollen etc. welche für die eigentliche Funktionalität von *i-engine* benötigt werden.

Application: In separaten Ordner sind jeweils sämtliche Objekte einer Applikation – public, health, business - abgelegt.

Kunde: Individuelle Hilfsobjekte und Stammdaten für den Kunden.

Basismodul

Der Inhalt dieses Ordners ist gemäss folgenden Standardvorgaben strukturiert:

Basismodule

B		Öffnen
		Ordner erzeugen
		Sichtbarkeit erteilen
		Eigenschaften bearbeiten
		Modul erzeugen
E		Zugriff erteilen

Ablagen

Die Dokument-Dateien werden üblicherweise in der Datenbank abgelegt. Aber auch andere techniken sind unterstützt.

B		Ordner erzeugen
		Öffnen
		Ablage erzeugen

Attributprofile

Attributprofile enthalten die Definitionen zum den Aufbau der Anzeigemasken.

B		Attributprofil erzeugen
		Öffnen
		Ordner erzeugen
E		Einfügen

Client Files

Hilfsdateien, welche auf der Arbeitsstation des Anwenders zur Unterstützung von *i-engine* Funktionalitäten benötigt werden.

B		Client Datei importieren
		Einfügen
		Öffnen
		Eigenschaften bearbeiten
E		Zugriff erteilen
		Umbenennen
		Definitiv löschen
		Sichtbarkeit erteilen

Mandanten

Verwaltung unabhängiger Organisationseinheiten auf der gleichen *i-engine* Installation.

B		Öffnen
		Eigenschaften bearbeiten
		Mandant erzeugen
E		Umbenennen
		Einfügen
		Zugriff erteilen

		Sichtbarkeit erteilen
		Löschen
W		zu Hauptordner

Objektklassen
Diese entsprechen den unterschiedlichen Datenobjektes, welche aus dem semantischen Datenmodell abgeleitet werden.

B		Öffnen
		Objektklasse erzeugen
		Ordner erzeugen
E		Definitiv löschen

Prozesse
Prozesse legten den Arbeitsfluss für eine bestimmte Objektklasse fest.

B		Prozess einfügen
		Prozess klonen
		Öffnen
		Ordner erzeugen
		Eigenschaften bearbeiten
		Formular-Prozess kopieren
E		Definitiv löschen
		Sichtbarkeit erteilen

Rollen
Eine Benutzerrolle definiert Aufgaben, Eigenschaften und vor allem Rechte eines Benutzers in *i-engine*.

B		Ordner erzeugen
		Öffnen

		Rolle erzeugen
E		Sichtbarkeit erteilen

SQL Statements
Verzeichnis zur Ablage aller im Rahmen des Customizing erstellten und referenzierten SQL-Statements.

B		Öffnen
		SQL-Statement erzeugen
		Einfügen
		Kopieren

Services
Kommunikationsdienste für den Datenaustausch mit Drittanwendungen.

B		Öffnen
		Service erzeugen

Spezielle, reservierte Objekte

StyleSheets
Individuelle Style Sheet css-Dateinen.

B		Öffnen
		Stylesheet erzeugen
		StyleSheet importieren
E		Zugriff erteilen

Sucher / Hilfssucher

Verwaltung der Sucher, Constraints und Reports.

B		Bericht erzeugen
		Einfügen
		Sucher erzeugen
		Öffnen
		Eigenschaften bearbeiten
E		Zugriff erteilen
		Sichtbarkeit erteilen
		Kopieren

System Watcher

B		Öffnen
		System Watcher erzeugen
E		Einfügen
		Sichtbarkeit erteilen

Systemeinstellungen

B		Öffnen
		Parameter erzeugen
		Eigenschaften bearbeiten

Thumbnail Typen

B		Öffnen
		Zugriff erteilen

		Thumbnail Typ erzeugen

Vorlagen
Objekte & Dateien die zur Erstellung von modul-spezifischer Objektklassen benötigt werden.

B		Öffnen
		Ordner erzeugen
		Eigenschaften bearbeiten
		Einfügen
		Kopieren
E		Definitiv löschen
		Zugriff erteilen
		Umbenennen
		Sichtbarkeit erteilen
W		Status: Hauptordner

Wertelisten
Kunden- oder anwendungstypische Auswahllisten werden als Wertelisten implementiert.

B		Öffnen
		Werteliste erzeugen
E		Kopieren
		Definitiv löschen

Module

Im Ordner «Applikation» sind ein oder mehrere «Modul» Ordner angelegt. Der Inhalt jedes «Modul» Ordners ist entsprechend dem Basismodul strukturiert. Häufig enthalten die Module auch kundenspezifische Objekte. Diese sind im Bereich «Kunde» unter dem jeweiligen Objekt-Ordner in einem eigenen Modul-Ordner abgelegt.

Generelle Struktur für Ablage von Modulen:

be-Module		business-engine
he-Module		health-engine
pe-Module		public-engine
	Modul xxx Modul yyy	
		Objektverzeichnisse wie für Basismodul
		Ablage aller generellen Objekte, welche keiner Anpassung bedürfen.
Kunde		
	Modul xxx Modul yyy	
		Objektverzeichnisse wie für Basismodul
		Ablage aller Objekte, welche für den Kunden parametrisiert werden müssen.

Kunde

Die Inhalte des «Kunde» Ordners ist sehr individuell. Die Unterordner «Stammdaten» und «Parametrisierung» sind aber jeweils Bestandteile der Kundendaten.

Genereller Aufbau der Verzeichnisse im Kundenbaum:

Kunde			
	Stammdaten		
		Alle Benutzer	Benutzer
		Organisation	Organisationseinheiten
		Ordnungssysteme	
		Vorlagen	
		Reports	
		Suchen	
	Parametrisierung		
		Struktur entsprechend dem Basismodul	
		Kunden spezifische Parametrisierungen	

Modul xxx
Modul yyy

Struktur entsprechend dem Basismodul

Kunden spezifische Parametrisierungen

Datenablage

In der Datenbank sind sämtliche Objekte und Eigenschaften sowie alle Methoden abgelegt. Die Methoden sind in der Oracle Programmiersprache PL/SQL codiert.

Die Datenbank kann auf einen Rechner gelegt werden, welcher keine Verbindung zum Internet aufweist. Damit ist jeder Zugriff von unerlaubter Seite verhindert und höchster Datenschutz gewährleistet.

i-engine unterstützt für die Ablage der Dokumentobjekte neben der normalen Datenbank-Speicherung weitere Speichertechniken:

- FTP-Server (externe Dateiablage)
- XDB via FTP (Datenbank)
- ORDDICOM Dicom-Bilder (Datenbank)
- BLOB (Datenbank)
- URI (lokales Filesystem oder NFS)
- SMB (Samba Fileshare)
- IBM DR550 (Archivmedia)

Wie kann die Leistung verbessert werden?

Datenbank

Die Oracle Datenbank zeichnet sich durch eine ausgezeichnete Skalierbarkeit bezüglich Durchsatz und Datenmengen aus. Die Datenbank kann durch Hinzufügen weiterer Speichermedien für die Verwaltung von fast beliebigen Datenmengen ausgebaut werden. Ebenso kann die Leistungsfähigkeit von Oracle durch Verteilung auf mehrere Systeme sehr einfach gesteigert werden.

FTP-Server

Auch der Datenserver, dessen Speicherplatz durch die FTP-Server Software verwaltet wird, kann für Speicherung fast beliebiger Datenmengen ausgelegt werden. Zur Leistungssteigerung können für Datenbank und Datenserver getrennte Systeme verwendet werden.

Eine weitere Steigerung ist durch die Vervielfachung des FTP-Servers erzielbar. Dabei muss sich der Web-Server im gleichen Netzwerk-Segment wie der FTP-Server befinden. Andernfalls muss zusätzlich auch ein weiterer Web-Server vorgesehen werden.

Durch die gute Skalierung können praktisch beliebig viele Dokumente mit der benötigten Leistung verwaltet werden.

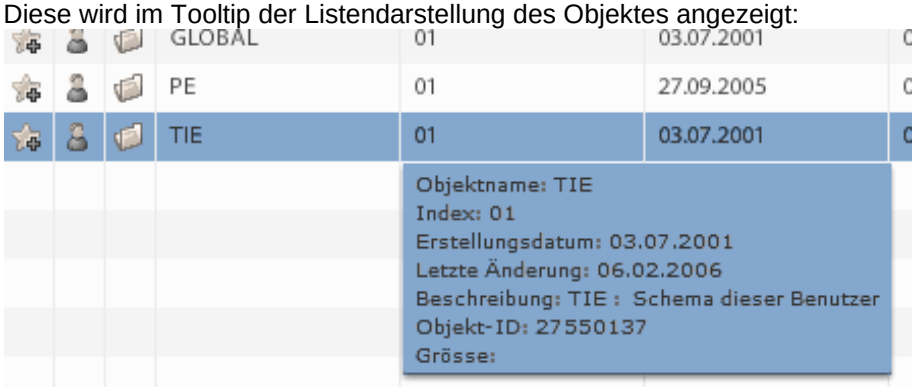
Generelle Arbeiten

Standard Operationen

Für das Customizing werden immer wieder interne Daten aus der *i-engine* Konfiguration benötigt. Hier eine Zusammenstellung wie die üblichen Werte bestimmt werden können.

Object-ID bestimmen

Diese wird im Tooltip der Listendarstellung des Objektes angezeigt:

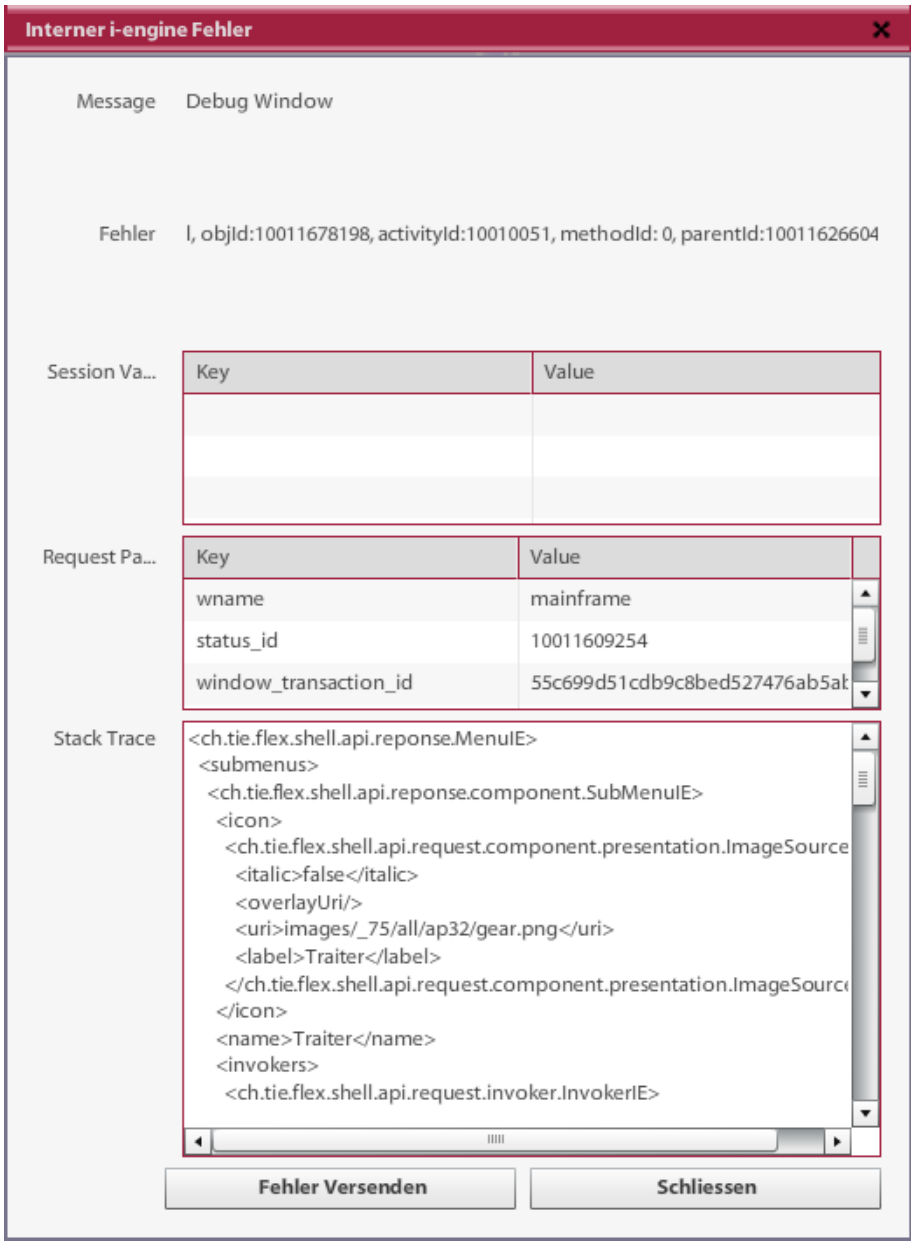


Zugeordneten Prozess bestimmen

1. «Object-ID» bestimmen
2. Tabelle OBJ_STATUS -> SELECT mit *OBJ_ID*=«object-id» => «STATUS_ID»
3. «Object-ID Sucher» verwenden -> mit «Status-ID» => zeigt Status
4. Über Kontextmenü **Ablage** erhält man den Prozess

Aktivitäts-Parameter bestimmen

Wenn die Aktivität mit gleichzeitig gedrückten Tasten <CTRL> + <SHIFT> ausgelöst wird, werden die an den Application Server gesandten Parameter angezeigt. Informationen wie Activity-ID, Methode-ID und Object-ID werden direkt angezeigt und können für die Problemanalyse eingesetzt werden.


Interner i-engine Fehler
Message
Debug Window
Fehler
l, objId:10011678198, activityId:10010051, methodId: 0, parentId:10011626604
Session Va...
Key
Value
Request Pa...
Key
Value
wname
mainframe
status_id
10011609254
window_transaction_id
55c699d51cdb9c8bed527476ab5ab
Stack Trace
<ch.tie.flex.shell.api.reponse.MenuIE>
<submenus>
<ch.tie.flex.shell.api.reponse.component.SubMenuIE>
<icon>
<ch.tie.flex.shell.api.request.component.presentation.ImageSource
<italic>false</italic>
<overlayUri/>
<uri>images/_75/all/ap32/gear.png</uri>
<label>Traiter</label>
</ch.tie.flex.shell.api.request.component.presentation.ImageSourc
</icon>
<name>Traiter</name>
<invokers>
<ch.tie.flex.shell.api.request.invoker.InvokerIE>
Fehler Versenden
Schliessen

Datenbank Standards

Synonyme und Views

xxx	Tabelle, enthält die Attribute für alle Sprachversionen
xxx_L_n	User-spezifische View auf Tabelle xxx, stellt jeweils nur 1 Sprache zur Verfügung.
sys_xxx_TBL	Synonym auf Tabelle xxx_TBL; es werden die Rechte des Benutzers berücksichtigt.
sys_xxxx	Synonym auf Tabelle xxx (default) resp. View xxx_Ln; abhängig von der aktuellen Spracheinstellung des Benutzers.

Die beiden Tabelle OBJECT_TBL, OBJCLASS_TBL sind bei der Bezeichnung eine Ausnahme von der Regel.

PL/SQL Funktionen und Prozeduren

Für die Bezeichnung von PL/SQL-Funktionen gilt folgende Vereinbarung:

F_	Standard-Funktionen der Basisprozesse und -funktionen
CF_	Custom: spezielle, kundenspezifische PL/SQL-Funktionen
TF_	Templates. Vorlagen für die Erstellung von kundenspezifischen Funktionen: TF_<SQL_Typ>_<Methode> bspw. TF_PA_SETSHOW
E8_	Methoden-Funktionen für Direktaufruf in Parameter-Statements. Versionen für *i-engine* Version vor V8 sind mit «E_» gekennzeichnet..
M_	Von (Java) Methoden benötigte PL/SQL-Funktionen
M8_	Spezielle Versionen für *i-engine* Version 8.
Ordner Datei	PLSQL/functions z.B. f_[name].sql

Für die Bezeichnung von PL/SQL-Prozeduren gilt folgende Vereinbarung:

P_ A_	Standard-Prozedur spezielle «Grant» Prozeduren
CP_	Custom: spezielle, kundenspezifische PL/SQL-Prozedur
TP_	Templates. Vorlagen für die Erstellung von kundenspezifischen

	Prozeduren: TP_<SQL_Typ>_<Methode> bspw. TP_PA_SETSHOW
Ordner Datei	PLSQL/procedures/ z.B. p_[name].sql

Packages

Für die Bezeichnung von Packages gilt folgende Vereinbarung:

PKG_	Basis
CPKG_	Custom: spezielle, kundenspezifische Packages
Datei	Der Package Code wird in die Dateien pkg_«name».sql (HEADER) und pkg_«name»_body.sql (BODY) aufgeteilt.
Beispiel	WFM_PACKAGE -> PKG _WFM URLKEY_PACKAGE -> PKG _URLKEY

Typen

Für die Bezeichnung von Typen gilt folgende Vereinbarung:

TY_	Basis
CTY_	Custom: spezielle, kundenspezifische Typen
Ordner	PLSQL/types/
Type: Object	TY_«NAME» as object(..) Beispiel: DICT_ENTRY -> TY_DICT Datei: ty_«name».sql
Type: VAR-RAY	TY_«OBJECT»_«N»_«NV» «OBJECT» soll dem Namen eines definierten PL/SQL Objekt (oben) entsprechen oder einem Standard DB Datentyp. «N» beschreibt z.B. die Grösse eines VARCHAR (VAR-CHAR2(100) oder VARCHAR2(1000)) «NV» entspricht der Grösse des VARRAY. Beispiel: ARRAY_OF_VARCHAR2_100 -> TY_VARCHAR2_1000_100 Datei: ty_«object»_«n»_«nv».sql

Type: TABLE	TY_«OBJECT»_TBL «OBJECT»: soll dem Namen eines definierten PL/SQL Objekt (oben) entsprechen oder einem Standard DB Datentyp. Beispiel: TABLE_OF_INTEGER_TYPE -> TY_INTEGER2_TBL Datei: ty_«object»_tbl.sql

Trigger

Für die Bezeichnung von Trigger gilt folgende Vereinbarung:

TRI_	Basis TRI_«COL»_«Type» «Type»: «U»pdate, «I»nsert, «D»elete. «COL»: Falls es mehrere Trigger gibt für eine Tabelle zusätzlich den Spaltenamen angeben In Oracle sind die Trigger unter der Tabelle angelegt - die Zuordnung folglich eindeutig. In der i-engine Ablage wird der Trigger unter dem Modul-Verzeichnis abgelegt und sollte mit dem Tabellennamen erweitert werden: TRI_«MODUL»_«COL»_«Type»
CTRI_	Custom: spezielle, kundenspezifische Trigger CTRI_«COL»_«Type»

Sequenzen

Für die Bezeichnung von Sequenzen gilt folgende Vereinbarung:

S_	Basis
CS_	Custom: spezielle, kundenspezifische Sequenzen

Jobs

Die Bezeichnung der Jobs wird nach folgender Konvention zusammengesetzt:

User AddRole	«USR_ARL»<user_id>_<role_id>
Object SetShow	«OBJ_SHO»<obj_id>_<role_id>
Status Notification	«STA_NTF»<obj_id>_<status_id>
Status Expiration	«STA_EXP»<obj_id>_<status_id>

Watch_Mail (expiration)	«WAT_MLE»<obj_id>_<status_id>
Watch_Mail (notification)	«WAT_MLN»<obj_id>_<status_id>
IPC	«J_IPC»<xxxx>

Hinweis:
<user_id>,<role_id>,<obj_id> und <status_id> werden jeweils als 11-stellige IDs angegeben.

Index

Für die Bezeichnung der Index gilt folgende Vereinbarung:

X_[name]	Normaler Index
T_[name]	Text Index
XB_[name]	Bitmap Index

Versionierung

Die Definition der folgenden DB-Objekte werden alle als SQL-Scripts gespeichert und unter Versionierung (CVS/SUBVERSION) gestellt.

- Funktion
- Prozedur
- Packages
- Typen (Object, VARRAY und Table)
- Trigger
- Sequenz

Die Definition der folgenden DB-Objekte wird nicht in SQL Scripts gespeichert.

- Synonym
- Tabellen
- View
- Index

Sprache des Codes (Variable-, Funktions-, Prozedurname usw.) ist immer Englisch.

Nomenklatur Oracle Bezeichner

Die vollständige Bezeichnung von PL/SQL Prozeduren und Funktionen wird nach folgendem Schema zusammengesetzt:

Applikation	«CF» Function «CP» Procedure «TF» Template Function «TP» Template Procedure
SQL-Statement Typ	AF: AFTER_STMT BC: BEFORE_STMT CB: CHECKBOX_STMT CS: CONDITIONAL_STMT CN: CONSTRAINT DE: DEFAULT_STMT DY: DYN_STMT EX: EXPIRATION_STMT FN: FINDER FK: FORK_STMT NX: NEXT_ACTIVITY_STMT PA: PARAMETER_STMT PO: POPUP RP: REPORT RT: RETURN_STMT RO: ROLE_STMT TA: TARGET_STMT VA: VALUE_STMT
Modul-Kurzbezeichnung	Die Modul-Kurzbezeichnung wird im Verzeichnisnamen der Modulablage festgehalten. bspw. «SPS - Spesenmodul» Die Modul-Kurzbezeichnung sollte max. 3 Zeichen umfassen.
Prozess-Kurzbezeichnung	Die Prozess-Kurzbezeichnung wird im Beschreibungsfeld des Prozesses festgehalten. bspw. «SPA - Spesenantrag» Die Prozess-Kurzbezeichnung sollte 3 Zeichen umfassen.
Status-Kurzbezeichnung s. Tabelle	Die Status-Kurzbezeichnung wird im Beschreibungsfeld des Prozessstatus festgehalten. bspw. «GEN - Genehmigen» Hier kann eine einfache Nummerierung der Status verwendet werden. Die Kurzbezeichnung sollte

	max. 3 Zeichen umfassen.
Aktivitäts-Kurzbezeichnung s. Tabelle	Als Kurzbezeichnung kann eine Teil des Methodennamens verwendet werden. Andernfalls ist die Aktivitäts-Kurzbezeichnung im Beschreibungsfeld der Aktivität festzuhalten. bspw.: «VISUM - zur Visierung» Kurzbezeichnung sollte max. 9 Zeichen umfassen.
Constraint-, Finder- oder Report-Kurzbezeichnung s. Tabelle	Die Kurzbezeichnung wird im Beschreibungsfeld festgehalten. Kurzbezeichnung sollte max. 12 Zeichen umfassen.
IPC-Kurzbezeichnung s. Tabelle	Die Kurzbezeichnung wird im Namensfeld festgehalten. bspw.: «TRIG - Scanning starten» Kurzbezeichnung sollte max. 9 Zeichen umfassen.
Attribut-Kurzbezeichnung s. Tabelle	Die Kurzbezeichnung wird im Beschreibungsfeld des Attributprofils festgehalten. Kurzbezeichnung sollte max. 12 Zeichen umfassen.

SQL Typ	Status	Aktivität	Finder	IPC	Attribut
AF	✘	✘			
BC	✘	✘			
CB					✘
CS	✘			✘	
CN			✘		
DE					✘
DY	✘	✘			
EX	✘				
FN			✘		
FK	✘				
NX	✘	✘			
PA	✘	✘			
PO					✘
RP			✘		
RT			✘		
RO	✘	✘			
TA	✘			✘	
VA					✘

Beispiele

CF_FK_SPS_SPA_GEN
CP_CN_APA_ORD_SCANSTR
CF_DE_DOT_PRZ_USERINPUT
CF_DY_SPS_SPA_GEN_FILLDATA

Hinweise

- Der Bezeichner darf maximal eine Länge von 27 Zeichen aufweisen (Einschränkung von Oracle)
- Oracle kann Bezeichner nur in Grossbuchstaben ablegen.
- Der Bezeichner wird bei der Erstverwendung vergeben und auch bei anderweitigem Einsatz beibehalten.
- Im PL/SQL Routinen Header sind alle Verwendungen auszuweisen.

- Für Templates wird der Bezeichner wie folgt aufgebaut:
  <TF>_<SQL Type>_<Methode>

SQL-Statements

i-engine ist eine Datenbank-basierende Anwendung. Das Customizing bedingt deshalb den virtuosen Umgang mit Tabellen und SQL-Abfragen. Kundenspezifische SQL-Statements liefern die notwendigen Ergebnisse in Attributprofilen und Prozessen. Die Struktur für die Ergebnisausgabe eines SQL-Statements ist für jeden Einsatzort festgelegt. Der korrekte Aufbau der Abfrage gehört zu den Kernaufgaben der Customizings.

Über SQL-Statements werden oft spezifische Objekt-IDs festverdrahtet. Dadurch werden Abhängigkeiten erzeugt, welche später beim Updaten oder Releasewechsel wesentliche Aufwände bereiten. Ein gutes Systemdesign ist unabdingbare Voraussetzung, um die Erstellung spezifischer SQL-Statements zu minimieren und damit den Wartungsaufwand einzuschränken.

Der Zugriff auf die Tabellen – und Funktionen & Prozeduren - erfolgt generell über Synonyme. Jeder Benutzer hat ein eigenes Datenbankschema, in welchem hauptsächlich diese Synonyme definiert sind. Für die Bezeichnung der Synonyme wird «sys_» dem Tabellennamen vorangestellt.

Durch die Verwendung von Synonymen in SQL-Statements werden

- die Texte gemäss aktueller Spracheinstellung geliefert

und

- die Sichtbarkeit auf Objekte ist berücksichtigt.

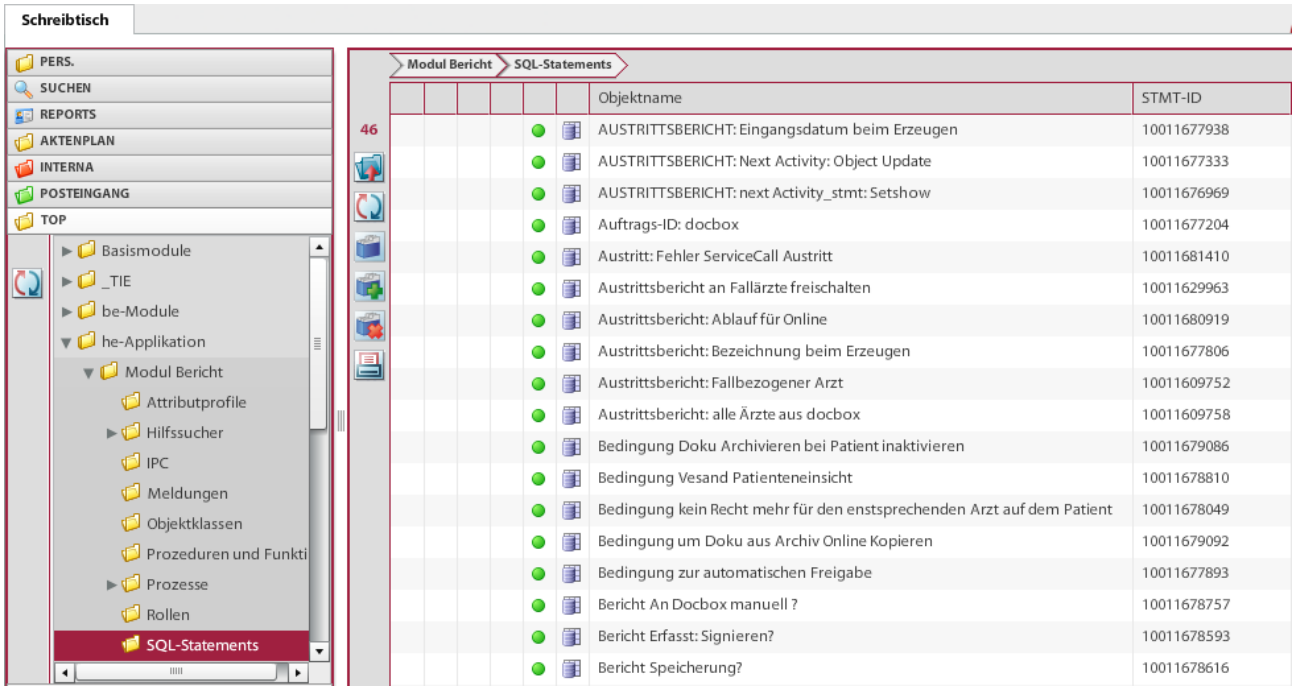
Hinweis

- Nur in Ausnahmefällen – Leistungsprobleme – sollte direkt auf die Tabellen zugegriffen werden. In diesem Fall muss die Sichtbarkeit selbst geprüft werden.
- Da die Synonyme von der Spracheinstellung abhängig sind, müssen diese bei einer Umstellung beim nächsten Login des Benutzers zuerst erstellt werden.
- Die Tabellen sind unter dem jeweiligen Kunden-Schema abgelegt. Andere Benutzer können auf die Tabellen nur zugreifen, wenn die Schema-Bezeichnung vorangestellt wird. Dadurch würde die Syntax der SQL-Befehle unnötig aufgebläht. Die Verwendung von Synonymen erlaubt den Zugriff auf die Tabellen über normale Tabellenbezeichnungen.

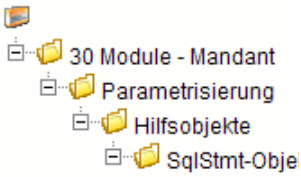
Erzeugen und Speichern

DB Action

Liegt das SQL-Statement vor, muss es korrekt in der *i-engine* Datenstruktur abgelegt werden. Teilweise müssen die notwendigen Schritte direkt in der Datenbank vorgenommen werden.



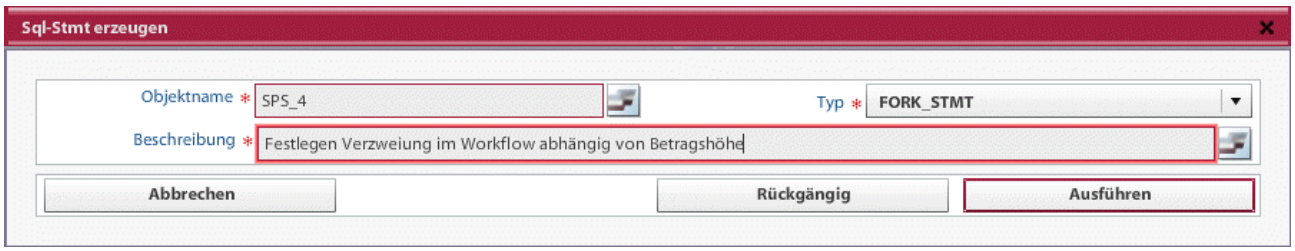
1. Ablageort SQL-Statement bestimmen



Alle SQL-Statements liegen im speziellen «SQL-Statements» Ordner. Dabei muss jener im korrekten Hauptordnerzweig verwendet werden. s. Abschnitt «*Ordnerstruktur* (Seite 13)».

2. Neues SQL-Statement Objekt erzeugen

Über das Kontextmenü **SQL-Statement Erzeugen** wird die Eingabemaske aufgerufen.



Der Objektname wird entsprechend der festgelegten Nomenklatur eingesetzt. Auf die Informationsteile bis *Modul-Mnemonic* kann verzichtet werden, da diese klar sind. Das Feld *Typ* definiert den Verwendungszweck der SQL-Abfrage und dient zur Filterung bei der späteren SQL-Statement Auswahl.

Über die Schaltfläche *Ausführen* wird ein neues Objekt erzeugt. Das neue SQL-Statement hat nun eine eindeutige *STMT_ID* erhalten. Diese wird für die folgenden Schritte benötigt.

3. Ablage des eigentlichen SQL-Statements

Das SQL-Statement selbst muss manuell in der Datenbanktabelle SQL_STMT abgelegt werden.

STMT_ID	Wird mit *STMT_ID* aufgesetzt
STMT_NAME	Wird mit *STMT_ID* aufgesetzt
DESCR	Entspricht dem Beschreibungsfeld aus Schritt 2
TYPE	Entspricht dem Feld *Typ* aus Schritt 2
SQL_STMT	SQL-Statement String

Hinweis

- Wird das SQL-Statements mittels eines SQL-Befehls im SQL_STMT Feld abgespeichert, müssen einzelne «'»durch «''» ersetzt werden.

4. Verwendung

Über den eindeutigen *STMT_ID* kann das SQL-Statement nun in Attributprofilen, Aktivitäten etc. eingesetzt werden.

Nomenklatur SQL-Statements

Die Bezeichnung für die SQL-Statements ist anlog zur Nomenklatur der Oracle Bezeichner aufgebaut. Da die SQL-Statements innerhalb des Moduls im Ordner «SQL-Statements» abgelegt werden, entfallen aber einige Elemente aus dem Oracle Bezeichner. Eigentlich bestehen keine Einschränkungen bzgl. Grosskleinschreibung oder Zeichenlänge. Der Übersichtlichkeitshalber werden aber die Bezeichner analog den Oracle Bezeichner mit den gleichen Mnemonics aufgebaut werden.

AF, BC, DY, NX, PA, RO
<Prozess>_<Status>_<Aktivität>

CB, DE,PO, VA
<Prozess>_<Attributprofil>

CS,TA
<Prozess>_<Status>_<IPC>

CN, FN
<Prozess>_<Finder>

EX, FK, RE, RT
<Prozess>_<Status>

SQL-Statement Typen

Folgende Werte sind für *Typ* vorgesehen:

AFTER_STMT	Über das *After-Statement* können nach Abschluss der Methode noch Schreiboperationen in der Datenbank vorgenommen werden. Wird in der Aktivität - via GUI - im Feld *After Statement* definiert. **After-Statements sind aus wartungstechnischen Gründen unzulässig. Sie dürfen nur zur Lösung ganz spezifischer Anforderungen eingesetzt werden.**
BEFORE_STMT	Das *Before-Statement* kann vor der Ausführung der Methode die Voraussetzung oder Datenbestände überprüfen. Wird in der Aktivität - via GUI - im Feld *Before Statement* definiert

CHECKBOX_STMT	Beim Popup wird eine Auswahlliste angezeigt. Das *Checkbox-Statement* definiert, welche Werte ein Auswahlkästchen erhalten. Wird in der Tabelle ATTRIBUTE_SHOW im Feld TAG_PREFS definiert.
CONDITIONAL_STMT	Liefert den Rückgabewert «YES» oder «NO», um das Ausführen von Aktivitäten oder den Versand von E-Mails auszulösen. Für E-Mails: Wird in der Tabelle WATCH_PROCESS im Feld CODITIONAL_STMT_ID definiert. Für IPC: Wird in der Tabelle IPC im Feld CONDITIONAL_STMT_ID definiert.
CONSTRAINT	*Constraint* liefert das Suchresultat für die Listenanzeige zurück. Das SQL-Statement wird in der Tabelle FINDER im Feld STMT_ID eingetragen.
DEFAULT_STMT	Über das *Default-Statement* kann der Vorgabewert für ein Feld im Attributprofil aufgesetzt werden. Wird in der Tabelle ATTRIBUTE_SHOW in den Feldern DEFAULT_VALUE /.._Lx definiert.
DYN_STMT	Für einige Methoden - object.execute - können die notwendigen Parameter über ein *Dyn-Statement* angeliefert werden. Wird auf der zweiten Methodenmaske im Feld *Parameter 1* aufgesetzt.
EXPIRATION_STMT	Bei Ablauf der Frist für das Ablaufdatum wird dieses Statement ausgelöst. Wird auf einem Status - via GUI - im Feld *Ablauf-Statement* aufgesetzt.
FINDER	*Finder* liefert das Suchresultat für die Listenanzeige zurück. Das SQL-Statement wird in der Tabelle FINDER im Feld STMT_ID eingetragen.
FORK_STMT	In einem Prozessstatus kann eine Bedingung definiert werden, welche einen Folgeaktivität auslöst, um den CONDITION-Prozessstatus zu verlassen. Wird im Prozessstatus - via GUI - im Feld *Fork-Stmt-ID* definiert.

NEXT_ACTIVITY_STMT	Nach Beendigung der aktuellen Aktivität kann ohne weitere Interaktion eine weitere Aktivität gestartet werden. Wird in der Aktivität - via GUI - im Feld *Auswahl nächste Aktivität* definiert.
PARAMETER_STMT	Das *Parameter-Statement* liefert die Parameterliste zurück, die für die Ausführung von Methoden wie *Distribute()* benötigt werden. Definiert in Tabelle OBJECT_PROPERTY: Feld VALUE mit KEY=PARAMETER_STMT_ID Feld OBJ_ID mit activity_ID aufgesetzt oder in Tabelle IPC: Feld TARGET_PARAMETER_STMT_ID
POPUP	Für die Eingabeelement Dropdown-Liste und Optionstaste werden die Anzeigewerte über ein *Popup-Statement* bestimmt. Dieser Typ wird auch für alle nicht speziell spezifizierten Anwendungsfälle eingesetzt. Wird in der Tabelle ATTRIBUTE_SHOW im Feld POPUP_OBJ_ID definiert.
REPORT	*Report* liefert die Datenauswahl für die Listenanzeige zurück.
RETURN_STMT	Die Auswahl im Sucherergebnis wird durch das *Return-Statement* für die Anzeige/Speicherung im Attributprofil aufbereitet. Wird beim Erzeugen des Suchers aufgesetzt; in der Tabelle FINDER im Feld RESULT_STMT_ID.
ROLE_STMT	Im Zeitpunkt der Ausführung des Watch-Prozesses kann die zu verwendende Rolle dynamisch bestimmt werden. Das Statement liefert eine oder mehrere Rollen-IDs. Rückgabewerte sind des SQL-Statements: Role_ID und Member_Type Eingetragen in der Tabelle WATCH_PROCESS im Feld ROLE_ID.
SYSTEM	Spezielle SQL-Statements, die Dropdown-Listen und Options-Auswahlen für Basisdienste implementieren.
TARGET_STMT	Liefert eine Liste der IDs für jene Objekte, für welche die Aktivität des IPCs ausgelöst werden soll.

	Ist in der Tabelle IPC im Feld TARGET abgelegt.
VALUE_STMT	Beim Popup wird u.U. der Rückgabewert nicht direkt zurückgeliefert, sondern über das *Value-Statement* bestimmt. Kann auch für die Aufbereitung der Anzeige bei Mehrfachwerten verwendet werden. Wird in der Tabelle ATTRIBUTE_SHOW im Feld VALUE_STMT_ID eingetragen.
WATCHER_PROCESS	Ein einziges Dummy SQL-Statement, welches zum Abonnieren der E-Mails verwendet wird (fixe ID:1) Wird in Tabelle WATCH_LIST referenziert.
WATCHER_SYSTEM	Einträge für alle "Nicht-objekt-orientierten" Meldungen. In «nicht standard» Fällen überprüft das *Watcher-System* SQL-Statement die Situation und generiert ggf. die Benachrichtigungsdaten. Keine weiteren Einträge notwendig.
WATCHER_ OBJSPECIFIC	Einträge für "objekt-orientierte" Meldungen. z.B. Änderungs-Überwachungs-Watcher.

Variablen

In den SQL-Statements können auch Variablen verwendet werden, welche zu Laufzeit aufgelöst werden. Es können die Felder einzelner Tabellen referenziert werden. Aktuell werden folgende Tabellen unterstützt:

?{object.xxxx}	Diese Felder sind für alle Objektklassen gemeinsam.
?{user.xxxx}	Objektklasse USER
?{patient.xxxx}	Objektklasse PATIENT
?{visit.xxxx}	Objektklasse VISIT
?{activity.xxxx}	Objektklasse ACTIVITY
?{parent.xxxx}	Zugriff auf die gemeinsamen Felder des übergeordneten Objektes
?{parent.parent.xxxx	Zugriff auf die gemeinsamen Felder des übergeordneten Objektes
?{root.xxxx}	Zugriff auf die gemeinsamen Felder des übergeordneten Root-Objektes

Zudem stehen folgende "errechnete" Variablen zur Verfügung:

?{object.next_wf_note}	
?{object.next_expiration_date}	
?{object.next_status_name}	
?{object.owner}	
?{object.doktyp}	
?{new_object_id}	Neu erzeugtes Objekt bei Create Methoden

Hinweis

- Variablen für Tabellenfelder werden nach der Syntax **?{xxxx}** aufgebaut.
- Variablen im Umfeld der Attributprofile sind hingegen nach der Syntax **?xxxx** aufgebaut.

Before-Statement

DB Action

Bevor die Aktivität ausgelöst wird, kann ein SQL-Befehl ausgeführt werden. Das SQL-Statement kann z.B. Randbedingungen für die Ausführung überprüfen. Das Ergebnis der Abfrage muss mindestens 1 Record mit folgendem Aufbau sein:

Rückgabe-Record

Status	Testergebnis «0»: ok «1»: Warnung -> die Aktivität kann mit *Weiter* ausgeführt werden «2»: Stopp -> es ist nur *Abbrechen* möglich «3»: Warnung + Aktivität -> zusätzlich Möglichkeit für Aktivität «4»: Stopp + Aktivität -> zusätzlich Möglichkeit für Aktivität «5»: Liste -> Anzeige von Objekten in einer weiteren Maske. s. Beispiel
Fehlermeldung	Entsprechender Meldungstext
Methoden-ID	Für die Behebung der Problems steht eine Aktivität direkt über eine Schaltfläche zur Verfügung (Status=«3» oder «4»). Die Auswahl der betreffenden Korrektur-Aktivität geschieht indirekt über die *Methoden-ID*. Im aktuellen Prozessstatus wird jene Aktivität ausgelöst, welche die betreffende Methode einsetzt. Nachdem die Aktivität durchgeführt wurde, wird der Before-Check noch einmal ausgeführt. Idealerweise ist das Problem nun behoben.
Object-ID	Objekt auf welches die angeführte Methode angewandt werden soll.
Parent-ID	Parent-Objekt zum betreffenden *Object-ID*
Referenz	Zur eindeutigen Unterscheidung, falls die Methode mehrfach eingesetzt ist.

Hinweis

- Wenn mehrere Fehlerbedingungen erfüllt sind, wird für jede ein Record zurückgegeben.
- Sollten mehrere Aktivitäten die gleiche Methode verwenden, wird zur eindeutigen Identifikation zusätzlich das *Referenz* Feld verwendet. Dh. In der Aktivität, welche den Before-Check auslöst und in der benötigten Korrektur-Aktivität müssen die Referenz-Werte übereinstimmen.
- Die Einträge für *Methoden-ID*, *Object-ID* und *Parent-ID* sind NULL für Status-Werte «1» oder «2».
- Falls das Before-Statement ausgeführt werden muss -Status<>«ok» - werden die Informationen mittels Standard-Methodenprofil ACTIVITY_EXECBEFORESTMT (ID: 10020301) angezeigt. Die Behandlung ist dabei speziell, da nur die BODY & BUTTON-Teile aus diesem Profil stammen.

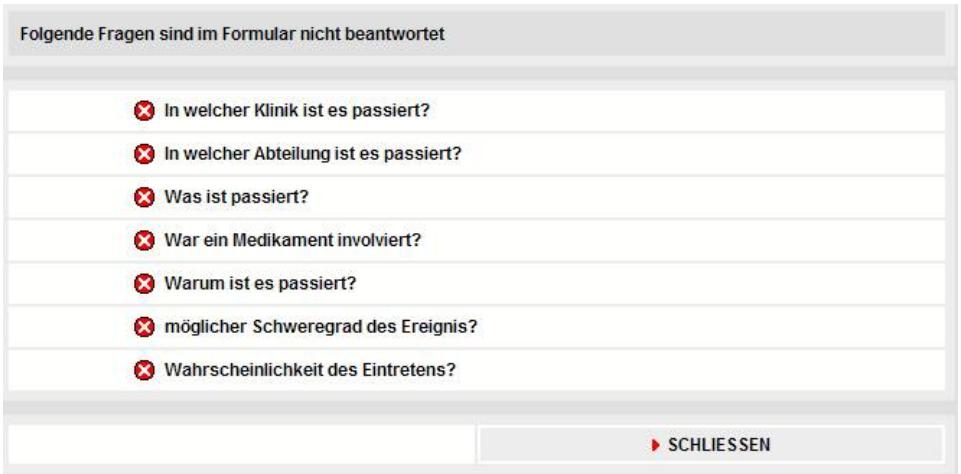
Für den HEADER Teil wird hingegen jener aus dem Attributprofil der aktuellen Methode eingesetzt.

Fehlermeldungen werden in Tabelle BOILERPLATE hinterlegt. Es sind folgende Einträge notwendig:

BOILERPLATE_TYPE	Fix «USER_MESSAGE»
DOMAIN	Fix «BEFORE_CHECKER»
TEXT TEXT_L1, L2, L3	Fehlermeldung ggf. in allen Sprachfassungen

Der Aufbau der Antwort-Records wird oft in eine PL/SQL-Funktion verlagert. In einem solchen Fall können auch Parameter – verwendete Form ist {0}, {1},.. – im abgelegten Textstring durch die PL/SQL-Funktion aufgelöst werden

Beispiel «Fehlermeldung»


Folgende Fragen sind im Formular nicht beantwortet
In welcher Klinik ist es passiert?
In welcher Abteilung ist es passiert?
Was ist passiert?
War ein Medikament involviert?
Warum ist es passiert?
möglicher Schweregrad des Ereignis?
Wahrscheinlichkeit des Eintretens?
SCHLIESSEN

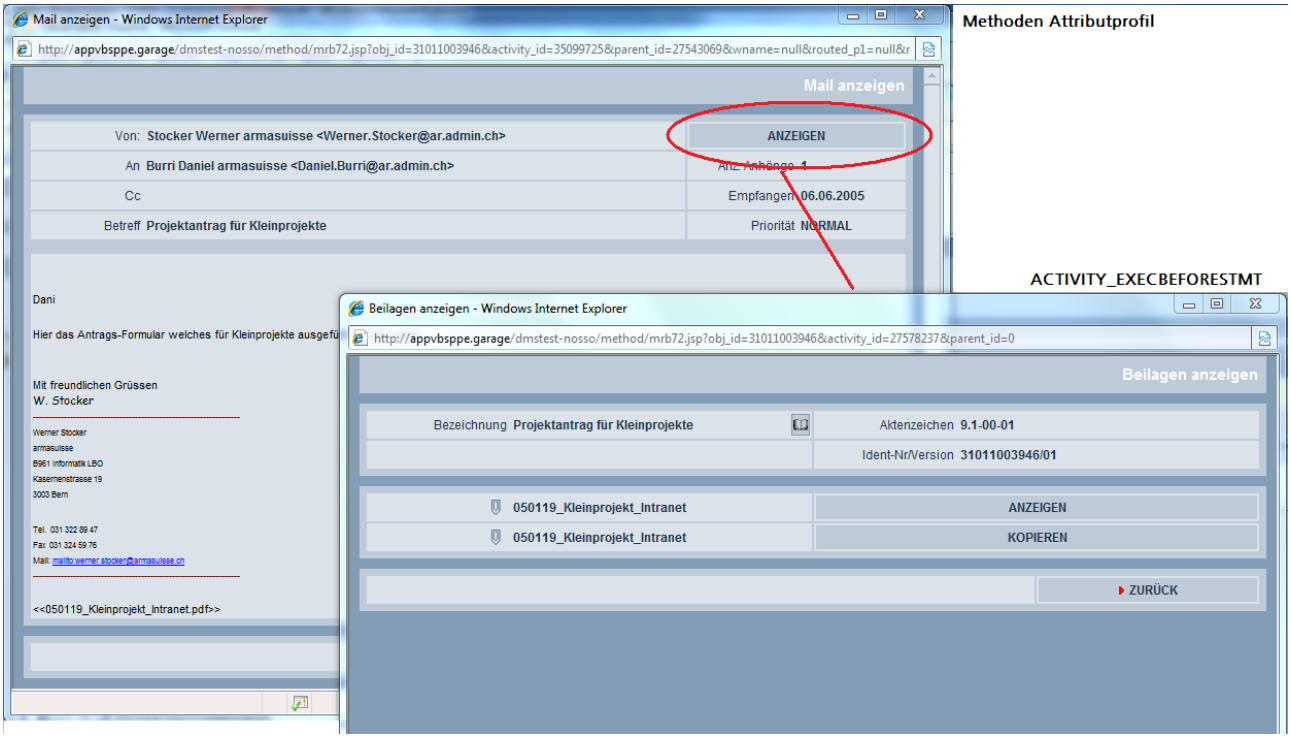
Prinzipieller Aufbau eines Before-Statements

```
CREATE FUNCTION CF_BC_SPS_SPA_AKT (obj_id) RETURN before-
list
BEGIN
      SELECT ….               // Abfragen der Bedingung

      IF condition=0 THEN
            out_rec.msg_status := 0
            out_rec.msg := 'OK'
      ELSE
            out_rec.msg_status := 2
            SELECT text INTO text_q FROM sys_boilerplate
                  WHERE boilerplate_id = 100065 // Mel-
dungsstext
            out_rec.msg := text_q
      END IF
      out_rec.obj_id := obj_id
      out_rec.methode_id := 0
      out_rec.parent_id := obj_id
      PIP ROW(out_rec)
```

```
      RETURN
END


CREATE TYPE before AS OBJECT (
      msg_status        INTEGER,
      msg               VARCHAR2(500),
      obj_id            INTEGER,
      methode_id        INTEGER,
      parent_id         INTEGER);
CREATE TYPE beforelist AS TABLE OF before;
```

Beispiel «Listenanzeige»

Mit der Rückgabe <Status=5> kann in einer weiteren Maske eine Liste von Objekten angezeigt werden. Eine Verwendungsmöglichkeit ist die kontrollierte Anzeige von E-Mail Anhängen, die selbst nicht von einem Geschäftsprozess gesteuert werden.



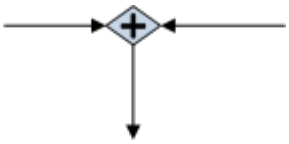
Expiration-Statement

Das Expiration-Statement liefert 2 Rückgabewerte:

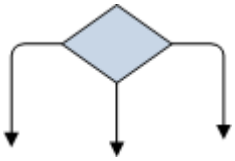
- ID der auszulösende Aktivität
- ID des dazu benötigten Parameter-Statements; dieses liefert die Parameter zur Methode der Aktivität

Bei der Auslösung der Aktivität wird noch einmal überprüft, ob die betreffende Aktivität im aktuellen Prozessstatus überhaupt noch verfügbar ist, sonst löscht sich der Job automatisch.

Fork-Statement



Auf einem Prozessstatus kann eine Bedingung - «Fork» - definiert werden, die unabhängige Prozesszweige mit einander synchronisiert. Das Fork-Statement wird typischerweise in einem eigenen Prozessstatus platziert. D.h. im betreffenden Prozessstatus sind keine weiteren Aktivitäten definiert, ausser einer einzigen zum Fortführen des Workflows (Distribute resp. Commit).



Ein solcher «Fork» kann auch verwendet werden, um einen oder mehrere Folgeprozess(e) auszulösen. Dies bedeutet, dass der Prozess auf mehrere unabhängige Threats aufgeteilt wird.

Das Fork-Statement ist ein SQL-Statement, welches bestimmte Bedingungen überprüft – z.B. sind alle Teilprozesse im aktuellen Prozessstatus angekommen – und im Erfolgsfall die ID der nächsten Aktivität definiert. Das Ergebnis der Abfrage muss genau 1 Record mit folgendem Aufbau sein:

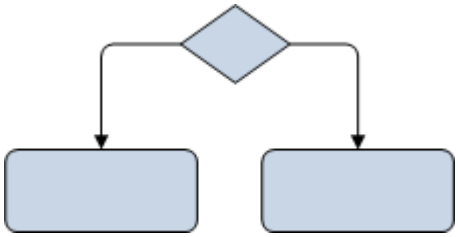
Rückgabe-Record

Aktivität ID	Wenn die Bedingungen erfüllt sind, wird die ID der Aktivität zurückgeliefert. Dies muss natürlich eine Aktivität sein, die im aktuellen Prozessstatus definiert wurde, und die Berechtigung zur Ausführung vorliegt.
Notiz	Entsprechender Meldungstext

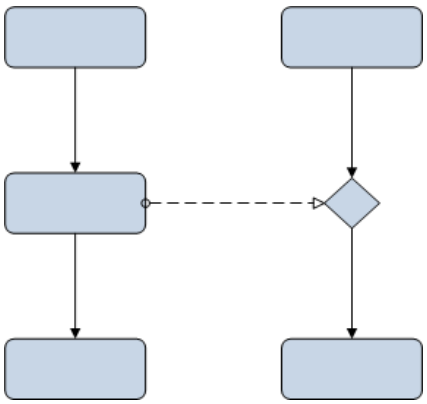
Hinweis

- Wird das Fork-Statement in einem Prozessstatus des Typ «DISTRIBUTION» eingesetzt, so muss als Ergebnis eine Aktivität mit der Methode *Object.Distribute()* resp. *Object.Commit()* verwendeten werden. Andere Methoden sind unzulässig.

- Die durch das Fork-Statement ausgelöste Aktivität muss im aktuellen Prozessstatus definiert sein – und sie kann sich auch nur auf das aktuelle Objekt beziehen.
- In Kombination mit einem Fork-Statement wird immer auch ein Parameter-Statement benötigt, welches die benötigten Parameter für die automatisch ausgelöste Aktivität aufsetzt.
- Das Fork-Statement kann auch mehrere Aktivitäten zurückliefern. Dadurch ist es möglich, mehrere unabhängige Prozesszweige zu starten.

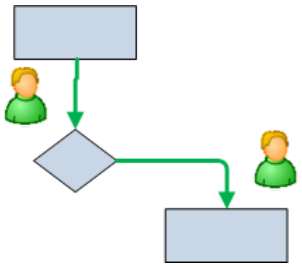


- Die Synchronisierung von zwei unabhängigen Prozessen wird ebenfalls über ein entsprechendes Fork-Statement gemacht. Im wartenden Prozess wird z.B. ein zustandabhängiges Attribut des anderen Prozesses regelmässig geprüft.

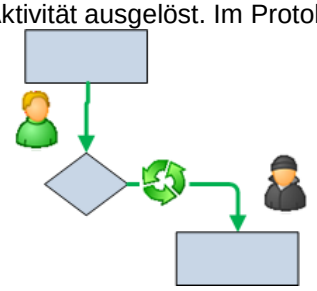


  Dieses Verfahren wird nur für Synchronisierungen innerhalb des gleichen Prozesses eingesetzt. Für die Synchronisation zwischen Prozesses ist die Verwendung der «Interprozess Kommunikation IPC» die bessere Lösung. In diesem Fall werden die notwendigen Methoden-Parameter über ein Parameter-Statement geliefert - und eine Eingabemaske wird folglich nicht angezeigt. Soll im Zielprozess aber eine Eingabemaske angezeigt werden, ist dies nur unter Verwendung der Methode *object.execute()* möglich.
- Das Fork Statement wird sofort nach Ende der Aktivität ausgewertet (Synchroner Fork). Kann die Bedingung für eine Prozessweiterführung nicht erfüllt werden, prüft ein Systemprozess in regelmässigen Zeitintervallen die Bedingung erneut (Asynchroner Fork).
- Für das Ausführen der vom Fork gemeldeten Aktivität muss eine entsprechende Berechtigung vorliegen.
  Es gilt:

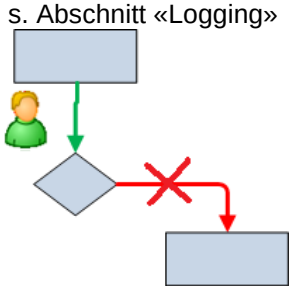
a) der aktuelle Benutzer hat die Berechtigung: die Aktivität wird ausgelöst. Im Protokoll wird der aktuelle Benutzer eingetragen.



b) der Systemmanager Dienst ist berechtigt: beim nächsten Lauf wird die Aktivität ausgelöst. Im Protokoll wird das System eingetragen.



c) das Objekt bleibt blockiert, es erfolgt ein entsprechender Eintrag im Logfile; s. Abschnitt «Logging»



Prinzipieller Aufbau eines Fork-Statements

```
CREATE FUNCTION CF_FK_SPS_SPA_CHECK (obj_id_p) RETURN
forklist
BEGIN
SELECT condition FROM sys_attribute
WHERE obj_id=obj_id_p
                                  // Abfragen der Bedingung

      IF condition=1 THEN
            out_rec.activity_id := 10011457850
            out_rec.wf_note := 'Abschluss'
      ELSE IF condition=2 THEN
            out_rec.activity_id := 10011457857
            out_rec.wf_note := 'In Validierung'
      END IF

      PIP ROW(out_rec)
      RETURN
END


CREATE TYPE fork AS OBJECT (
      activity_id INTEGER,
      wf_note           VARCHAR2(500));
CREATE TYPE forklist AS TABLE OF fork;
```

Eigentliches FORK SQL-Statement

```
SELECT activity_id, wf_note FROM
table(C_FK_example(?obj_id))
```

Parameter-Statement

Methoden können in einigen anderen Methoden programmgesteuert ausgelöst werden. Im normalen Geschäftsprozess erhalten die Methoden die erforderlichen Parameter über Werte aus dem Attributprofil. Bei einer programmgesteuerten Verwendung müssen diese Parameter anderweitig zur Verfügung gestellt werden. Dazu dient das Parameter-Statement

Beispiel

Wenn ein Fork-Statement verwendet wird, löst dieses üblicherweise eine Aktivität mit der Methode *Distribute()* aus. Dieser Methode müssen die notwendigen Parameter (z.B. Rollen, Mitglieder,..) übergeben werden. Ein spezielles SQL-Statement liefert diese als Rückgabewerte zuhanden der Methode.

Die Verknüpfung zwischen Aktivität und Parameterliste aus einem SQL-Statement wird über die Tabelle OBEJCT_PROPERTY vorgenommen.

Feld	Bedeutung
OBJ_ID	ID der Aktivität, welche die Methode enthält
KEY	Fix: «PARAMETER_STMT_ID»
VALUE	ID des SQL-Statements, welches die Parameterliste zurückliefert.

Die notwendigen Parameter sind für jede Methode individuell.

Das Parameter-Statement muss die erforderlichen Parameter als Tabelle in der Form «Feldname» / «Wert» liefern. Die Struktur der Tabelle ist von der Methoden abhängig. Um die Erstellung zu vereinfachen, liegen für oft verwendete Aufrufe die Parameter-Statements als PL/SQL Vorlagen vor:

TF_PA_DISTRIBUTE	Automatisierter Distribute im FORK
TF_PA_SETSHOW	Automatisierte Änderungen der Sichtbarkeit über einen IPC
TF_PA_MOVE	Automatisiertes Verschieben durch eine IPC ausgelöst

Für den Anwendungsfall wird mit Hilfe der Vorlage eine entsprechende Funktion erstellt. Die Bezeichnung wird entsprechend der Nomenklatur-Definition gesetzt. z.B. «CF_PA_SPS_SPA_4_DOIT».

Für den Aufbau einer Zeile der Rückgabetabelle stehen – abhängig vom Parametertyp – entsprechende Funktionen zur Verfügung:

- Nummerisch: ty_param_n (Parametername, Wert)
- Zeichen: ty_param_v(Parametername, Wert)
- Datum: ty_param_d(Parametername, Wert)

Beispiel Distribution TF_PA_DISTRIBUTE:

```
function TF_PA_Distribute (
         obj_id_p      in integer ,
         activity_id_p in integer ,
         parent_id_p   in integer )
  return ty_param_tbl authid current_user
is
--=============================================================================================
--Name der Funktion:        TF_PA_Distribute
--Beschreibung:
--Eingabeparameter:         obj_id, activity_id, parent_id
--Ausgabeparameter:         alle für eine Object.Distribute()
--Aufruf: select TF_PA_Distribute
                  (?{object.obj_id}, ?{activity.activity_id}, ?{parent.obj_id}) from dual
--Ersteller:               the-i-engineers
--Erstellungsdatum:        03.Februar 2011
--Aenderungsdatum:
--=============================================================================================

-- Data defintions
obj_id_q integer;
s_to_chosen_q varchar2(1000);
s_cc_chosen_q varchar2(1000);
wf_note_q varchar2(1000);
next_notification_date_q date default null;
next_expiration_date_q date;
followup_q                  varchar2(10) default 'NO';
action_q          varchar2(50) default 'Distribute';
summary_member_to_q         varchar2(1000) default null;
summary_member_cc_q         varchar2(1000) default null;
summary_member_fu_q         varchar2(1000) default null;
summary_distributor_q varchar2(1000) default null;
act_reference_q    varchar2(200);
Status_reference_q varchar2(200);

begin
         --p_autonomous_logger('TP_PA_Distribute','Start' ,'--','--' ,'--',
                  obj_id_p, activity_id_p, parent_id_p,'TEST' );


  --s_to_chosen => wenn mehrere Personen/ Rollen ist die
  --      Syntax = s_to_chosen_q := role_1||','||role_2;
          Select role_id into s_to_chosen_q
          from sys_role
          where role_name = 'Spesengenehmiger';

  --s_cc_chosen=> wenn mehrere Personen/ Rollen ist die
  --      Syntax = s_to_chosen_q := role_1||','||role_2;
  s_cc_chosen_q := '';

  --wf_note   ---------
  -- ist bereits oben behandelt

   --next_notification_date --bereits auf null

  --next_expiration_date => muss immer ausgefüllt sein!!
  select expiration_date
```

```
        into next_expiration_date_q
        from sys_obj_status
        where obj_id = obj_id_p;
  if next_expiration_date_q is null then
        next_expiration_date_q  := sysdate + 1;
  end if;

  --followup --bereits auf 'NO'  ----------
  --action --bereits auf 'Distribute'   --------
  --summary_member_to   -------------------
  select text
        into summary_member_to_q
        from boilerplate
   where boilerplate_id = 10011609220;

  --summary_member_cc   -------------------
  select text
        into summary_member_cc_q
        from boilerplate
   where boilerplate_id = 10106;
  --summary_member_fu
  -------------------
  select text
        into summary_member_fu_q
        from boilerplate
        where boilerplate_id = 10107;

  --summary_distributor   ---------------------
  select text
        into summary_distributor_q
        from boilerplate
        where boilerplate_id = 10108;


  -- ACHTUNG: obj_id, activity_id, parent_id sind immer gegeben und müssen in
                  Parameter nicht nochmals vorkommen.
  return ty_param_tbl ( ty_param_v('s_to_chosen', s_to_chosen_q),
        ty_param_v('s_cc_chosen', s_cc_chosen_q),
        ty_param_v('wf_note', wf_note_q),
        ty_param_d('next_notification_date', next_notification_date_q),
        ty_param_d('next_expiration_date', next_expiration_date_q),
        ty_param_v('followup', followup_q),
        ty_param_v('action', action_q),
        ty_param_v('summary_member_to', summary_member_to_q),
        ty_param_v('summary_member_cc', summary_member_cc_q),
        ty_param_v('summary_member_fu', summary_member_fu_q),
        ty_param_v('summary_distributor', summary_distributor_q) );
end;
```

Austesten eines Parameter-Statements

Parameter-Statements sind oft sehr in den Workflow verwoben und ein Austesten gestaltet sich entsprechend aufwändig. Die mit «P_Autonomous_Logger» aufgezeichneten Informationen helfen oft nicht weiter. Häufig werden die SQL/PL Aufrufparameter doch nicht so aufgelöst, wie gedacht.

In diesem Fall ist wie folgt vorzugehen:

- Der Aufruf der p8_exec Prozedur, welche das Parameter-Statement verwendet, ist im Logfile verfügbar.
  bspw.: *exec p8_exec(10031228660, 10011675826, 0, 's_to_chosen=10015676208,10015789368,.....*

- Der Aufruf wird direkt im Oracle SQL-Design ausgetestet. Die spezielle Routine «pkg_param.from_string» übernimmt die Aufbereitung der Parameter - identisch wie im Workflow.
  bspw.: *exec p8_exec(10031228660, 10011675826, 0,* **pkg_param.from_string***('s_to_chosen=10015676208,....*

Finder & Constraint-Statement

Das SQL-Statement vom Type FINDER resp. CONSTARINT liefert das Ergebnis einer Such- oder Filteranfrage zur Anzeige zurück. Das SQL-Statement besteht aus einigen fixen Teilen, wird in der Praxis aber stark den Erfordernissen der Anwendung angepasst.

Prinzipieller Aufbau eines Finder/Constraint-Statements

```
SELECT
      sys_object.objid,
      sys_objclass.super_img_collapsed,
      sys_objclass.objclass_name,
      ss.type,
      sys_object.classification     // Achtung kein Komma
      ?attribList,
      sys_objclass.sort_level
FROM
      sys_object, // aus Effizienzgründen sys_object_tbl
      sys_attribute,
      sys_objclass,
      sys_structure ss,
      sys_role                      // meist auch
WHERE
      sys_object.obj_id=sys_attribute.obj_id AND
      sys_object.obj_id=sys_structure.lower AND
      sys_object.objclass_id=sys_objclass.objclass_id AND
      sys_role.role_id(+)=sys_object.obj_id AND
      Suchkriterium
```

Default-Statement

Beim Erzeugen eines neuen Objektes kann der Vorgabewert in der Eingabemaske vorbesetzt sein. Ein entsprechendes SQL-Statement liefert den Vorgabewert.

In der Tabelle ATTRIBUTE_SHOW wird im Feld DEFAULT_VALUE die betreffende SQL-Statement ID eingetragen, welche den dynamisch bestimmten Vorgabewert zurückliefert. Für die Eingabe in diesem Feld gilt folgende Syntax:

- ?nnn: SQL-Statement ID
- ?clear: im Feld wird evt. vorhandener Wert aus der Systemvorlage gelöscht
- ?today: aktuelles Datum
- ?now: aktuelles Datum und Zeit
- String: fixer Wert

Bei existierenden Objekten kann der Defaultwert nur für Temporärfelder aufgesetzt werden.

Hinweis

- Das Default-Statement wird nur in den "Create"-Methoden und der "Commit" Methode ausgewertet.
- Typischerweise müssen deshalb für die Commit-Methode separate Attributprofile zur Verfügung gestellt werden. Eine «?clear» Anweisung, wie sie oft in Attributprofilen für "Create"-Methoden eingesetzt wird, hat bei der "Commit"-Methode unerwünschte Effekte.

Value-Statement

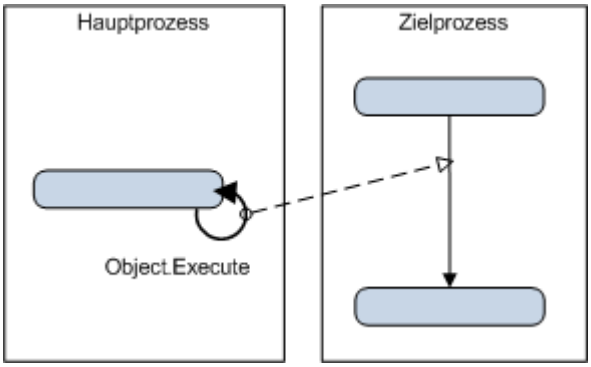
Bei der Verwendung einer Maske werden zuerst die Daten abgefragt. Bei neuen Objekten können fehlende Daten über entsprechende Default-Statements aufgesetzt werden. Bevor die Maske angezeigt wird, werden die Datenwerte u.U. durch Value-Statements in Anzeigewerte umgewandelt.

Das Value-Statement liefert immer ein Wertpaar Objekt-ID/Anzeigetext zurück. Je nach Anwendungsfall ist die Rückgabe mehrerer Wertpaare möglich.

Das Value-Statement wird ebenfalls bei Pop-up Fenstern zur Umrechnung der Übergabewerte verwendet.

Dyn-Statement

Das Fork-Statement kann nur Aktivitäten im aktuellen Status für das aktuelle Objekt auslösen. Über ein DYN SQL-Statement kann eine Aktivität bei einem beliebigen Objekt ausgelöst werden.



Die Methode *Object.Execute()* benötigt drei Parameter:

Parameter 1	Zielobjekt
Parameter 2	Auszuführende Aktivität – muss im aktuellen Prozessstatus des Zielobjektes natürlich zur Verfügung stehen.
Parameter 3	Parent-Objekt – falls von der Methode in der Aktivität benötigt.

Statt diese drei Parameter beim Definieren der Aktivität fest vorzugeben, kann anstatt des ersten Parameters der ID eines SQL-Statements angegeben werden. Das SQL-Statement muss dann diese drei Parameter zurückliefern.

Auf der Methodenmaske muss in diesem Fall das Feld *Executer Typ* den Wert «DYN_ACTIVITY» aufweisen.

Hinweis

- Im Unterschied zu Prozesssynchronisationen mit Fork oder IPC wird die Dialogmaske der Aktivität des Zielprozesses dem Benutzer angezeigt.

PL/SQL-Funktionen

PL/SQL verbindet die Abfragesprache SQL mit dem Komfort prozeduraler Programmiersprachen. Damit lassen sich komplexere Abfragen formulieren und austesten.

Über PL/SQL-Funktionen werden oft spezifische Objekt-IDs festverdrahtet. Dadurch werden Abhängigkeiten erzeugt, welche später beim Updaten oder Releasewechsel wesentliche Aufwände bereiten. Ein gutes Systemdesign ist unabdingbare Voraussetzung, um die Erstellung spezifischer PL/SQL-Funktionen zu minimieren und damit den Wartungsaufwand zu reduzieren.

In PL/SQL-Routinen dürfen keine Objekt-IDs verwendet werden. Ausnahme sind einzig jene von Standard Objekten (sog. *10-Mio Objekte*).

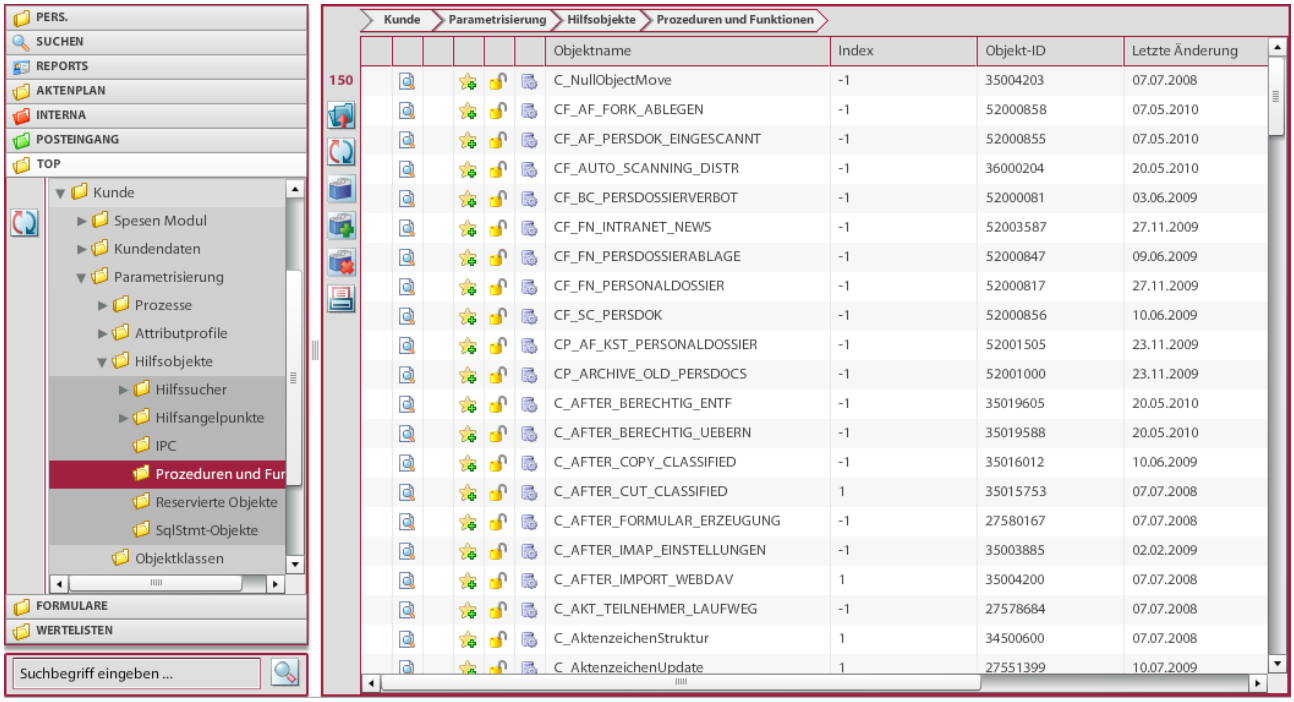
Nomenklatur PL/SQL-Funktionen

Der Objektname für die Ablage im Ordner «Prozeduren und Funktionen» wird identisch zum eigentlich Oracle-Bezeichner aufgebaut. s. entsprechenden Abschnitt.

Speichern und Verwenden

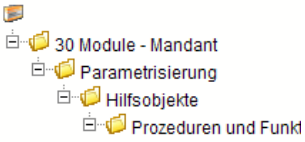
DB Action

Liegt die PL/SQL-Funktion vor, muss sie in *i-engine* korrekt abgelegt werden. Teilweise müssen die notwendigen Schritte direkt in der Datenbank vorgenommen werden.



PL/SQL-Funktion speichern

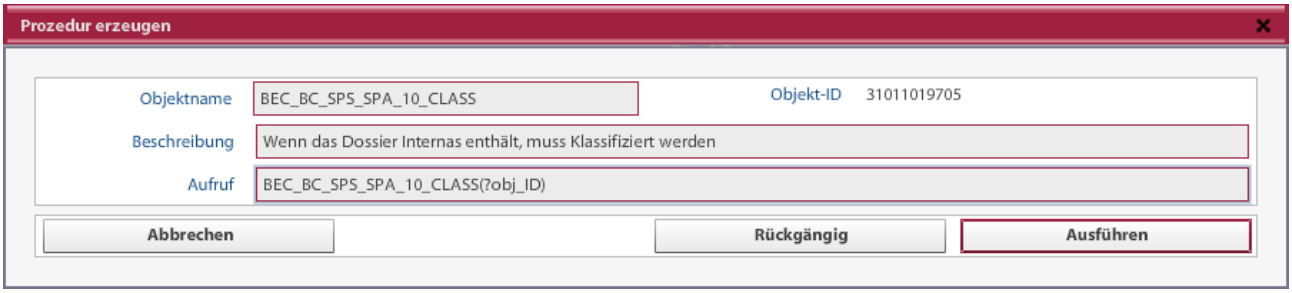
1. Ablageort PL/SQL-Funktion bestimmen



Alle PL/SQL-Funktionen liegen im speziellen «Prozeduren und Funktionen» Ordner. Dabei muss jener im korrekten Hauptordnerzweig verwendet werden. s. Abschnitt «*Ablagestruktur* (Seite 238)».

2. Neues PL/SQL-Funktion Objekt erzeugen

Über das Kontextmenü **Prozedur Erzeugen** wird die Eingabemaske aufgerufen:



Folgende Eingaben sind notwendig:

Objektname	Der Bezeichner ist durch die Nomenklatur festgelegt.
Beschreibung	
Aufruf	Festhalten der Aufrufparameter
Object-ID	Pro Mandant wird beim Aufsetzten des System ein Nummernbereich zugewiesen. Die neue *Objekt-ID* ist also Mandanten-abhängig.

3. Speichern der PL/SQL-Funktion

Die Funktion wird in einem DB-Tool entwickelt und ausgetestet. Über das Kontextmenü **Bearbeiten** kann – bei richtiger Zuordnung – dieses Werkzeug aufgerufen und die Funktion editiert und abgespeichert werden.

Für PL/SQL-Funktionen wird auch eine Versionierung unterstützt.

Verwendung

PL/SQL-Funktionen können überall eingesetzt werden, wo ein SQL-Statement vorgesehen ist. Das eingesetzte SQL-Statement ruft dazu die entsprechende PL/SQL-Funktion auf.

BEFORE_CHECKER SQL-Statement:

```
SELECT
      msg_status, msg, obj_id,
      decode(method_id,0,null,method_id), parent_Id
FROM
      Table(C_BC_SPS_SPA_3_DOIT(?obj_id,?parent_id ))
```

Werteliste erzeugen

In Dialogmasken werden immer wieder Auswahllisten eingesetzt. Statische, vom Basissystem vorgegebene Listen werden durch Wert aus der P_CODE Tabelle erzeugt. Kunden- oder anwendungstypische Listen werden über «Wertelisten» implementiert. Diese bieten den Vorteil, dass sie vom Anwender ohne direkte Datenbank-Manipulation modifiziert und erweitert werden können.

Definieren einer Wertetabelle

Der Ordner «Wertelisten» stellt die Funktionen für die Erstellung und Verwaltung von Wertelisten zur Verfügung. Über das Kontextmenü **Wertetabelle erzeugen** wird ein neuer Eintrag erstellt. Jeder benötigte Eintrag wird nun mit **Wert erzeugen** angelegt. Dabei kann auch die Anzeigenreihenfolge der Werte bereits festgelegt werden.

Kunde › Kundendaten › Stammdaten › Wertelisten › Dokumentenstati

	Sequenz	Bezeichnung	Beschreibung	Letzte Ände
11	0	---	leer	27.01.2009
	1	Freigegeben	Publiziert, genehmigt, abgelehnt	27.01.2009
	2	Nicht freigegeben	Alle Stati ausser 'Freigegeben'	27.01.2009
	3	Ausgecheckt	Alle Transition-Stati	27.01.2009
	4	Entwurf	Entwurf und Entwurf gesperrt	27.01.2009
	5	Stellungnahme	Stellungnahme	27.01.2009
	6	Bearbeitung	Bearbeitung und Bearbeitung gesperrt	27.01.2009
	7	Laufweg	Laufweg und Laufweg gesperrt	27.01.2009
	8	Archiviert im Dossier	Publiziert archiviert, genehmigt archiviert, abgelehnt archiv	27.01.2009
	9	Nicht archiviert im Dossier	Alle Stati ausser 'Archiviert im Dossier'	02.02.2009
	10	Vorregistriert	Vorregistriert	24.02.2009

Hinweis

- Nicht mehr verwendete Werte können nicht einfach gelöscht werden, da sie u.U. noch referenziert werden. Der Wertelisten-Prozess unterstützt deshalb auch die Deaktivierung der Einträge. Sie stehen dann zwar für die Anzeige noch zur Verfügung, können aber für Neueingaben unterdrückt werden.

Verwendung der Wertetabelle

Die Wertelisten werden in den Attributprofilen referenziert. Im Attributprofil wird eine Auswahlliste durch einen SQL SELECT-Befehl realisiert, der in zwei Spal-

ten den Wert und die Bezeichnung zur Anzeige zurückliefern muss. Prinzipiell sind in einem Attributprofil Eintrag folgende Felder aufzusetzen, um die Werteliste zu verwenden:

POPUP_TYPE: «POPUP»
POPUP_OBJID: ID des SQL-Statements
POPUP_MANDATORY: «YES»/«NO» entsprechend den Anforderungen
TAG: «SELECT»

Da die einzelnen Werte der Wertetabelle nichts anderes als Objekte sind, wurden sie in der SYS_OBJECT_TBL abgelegt. Die Abfrage holt sich alle Werte, welche die Werteliste als Parent-Objekt haben.

Beispiel:

```
SELECT obj_id, obj_name
FROM
      sys_object_tbl, sys_structure ss
WHERE
      ss.upper = 27409925 AND ss.lower = obj_id AND
      ss.type = 'ORIGINAL'
ORDER BY obj_name;
```

Der ID 27409925 entspricht hier der Objekt-ID der betreffenden Werteliste.

Diese Abfrage muss ggf. erweitert werden, so dass die aktuelle Sichtbarkeit der Werte, der Aktivstatus, die Sortierung oder fehlende Einträge berücksichtigt werden.

Beispiel:

```
SELECT o.obj_id, o.obj_name
FROM sys_object_tbl o, sys_structure, sys_obj_status s,
sys_attribute a
WHERE upper = 27409925 AND lower = o.obj_id
      AND o.obj_id = s.obj_id AND s.status_id = 27408844
      AND o.obj_id = a.obj_id AND a.page_nr = 0
ORDER BY a.n1;
```

Der ID 27408844 entspricht hier dem Status, welcher der «aktive Wert» in einer Werteliste einnehmen.

Hinweis

- Der SQL muss oft noch weitere Randbedingungen berücksichtigen. Bei einer reinen Anzeige, sollen auch abgelaufenen Einträge angezeigt werden; hingegen beim Neuerstellen sind nur die aktiven Werte anzubieten.
- Auch ein Vorgabewert oder Leerwert muss meistens möglich sein. Dazu muss das SQL-Statement natürlich entsprechend erweitert werden (UNION).

Speichern des SQL-Statements

Das SQL-Statement selbst muss entsprechend den Hinweisen im Abschnitt «*SQL-Statements -> Erzeugen und Abspeichern* (Seite 34)» abgelegt werden.

- als Typ wird «POPUP» festgelegt.

Logging

Das Basissystem der *i-engine* macht bei bestimmten Systemereignissen – wie Oracle oder WebDAV Fehlern – automatisch einen Log-Eintrag.

Für die Fehlersuche und das Debugging neuer Module können zusätzliche Log-Einträge erzeugt werden. Dazu steht die PL/SQL-Prozedur «P_Autonomous_Logger» zur Verfügung. Die Einträge werden in der Tabelle LOG eingetragen.

Aufrufparameter:

Parameter	Bedeutung
Typ	Name der Prozedur oder Funktion, welche das Logging aufruf.
msg_p	Meldungstext
original_msg	Ursprüngliche Fehlermeldung; bswp. Oracle Fehlertext
reference reference_p	2 Textfelder für Zusatzinfos
severity	0 «TRACE» (nur TIE intern) 1 «DEBUG» (nur TIE intern) 2 «INFORMATION» 3 «WARNING» 4 «ERROR» 5 «FATAL»

Hinweise

- Standard Log-Level ist 2, d.h. alle Meldungen ab Level «INFORMATION» werden tatsächlich geloggt, «TRACE» und «DEBUG» hingegen werden ignoriert/übersprungen.
- Der minimale Log-Level wird in der Tabelle PROPERTIES mit dem Eintrag LOGLEVEL festgelegt. Dieser wird generell für SYSTEM aufgesetzt, kann aber auch individuell für bestimmte USER definiert werden.
- Während einer Fehlersuche kann Log-Level temporär auf «DEBUG» oder bei Bedarf sogar «TRACE» gestellt werden. ACHTUNG: Es werden sehr viele Log-Einträge produziert, vor allem auf Level «TRACE». Dies führt zu grossem Platzbedarf auf DB sowie spürbar schlechterer System-Performance!
- Alle Einträge, welche mit der Funktion «p_autonomous_logger» erzeugt werden, werden im Log aufgezeichnet. D.h. keine Filterung nach Log-Level.

Deshalb können für diese Einträge auch beliebige Severity Textstrings verwendet werden. bspw. «TEST».

Beispiel:

p_autonomous_logger('TP_PA_Distribute','Start' ,'--','--' ,'--', obj_id_p, activity_id_p, parent_id_p,'TEST')

Auswertung

Für das Auswerten der Log-Einträge wird der «Report Error-Log» verwendet.

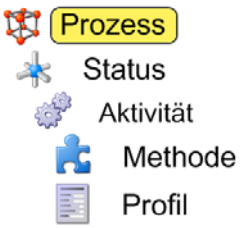
Hinweis

- In den Feldauswahllisten werden nur vorhandene Werte aufgeführt
- die Auswertung steht nur Administratoren zur Verfügung

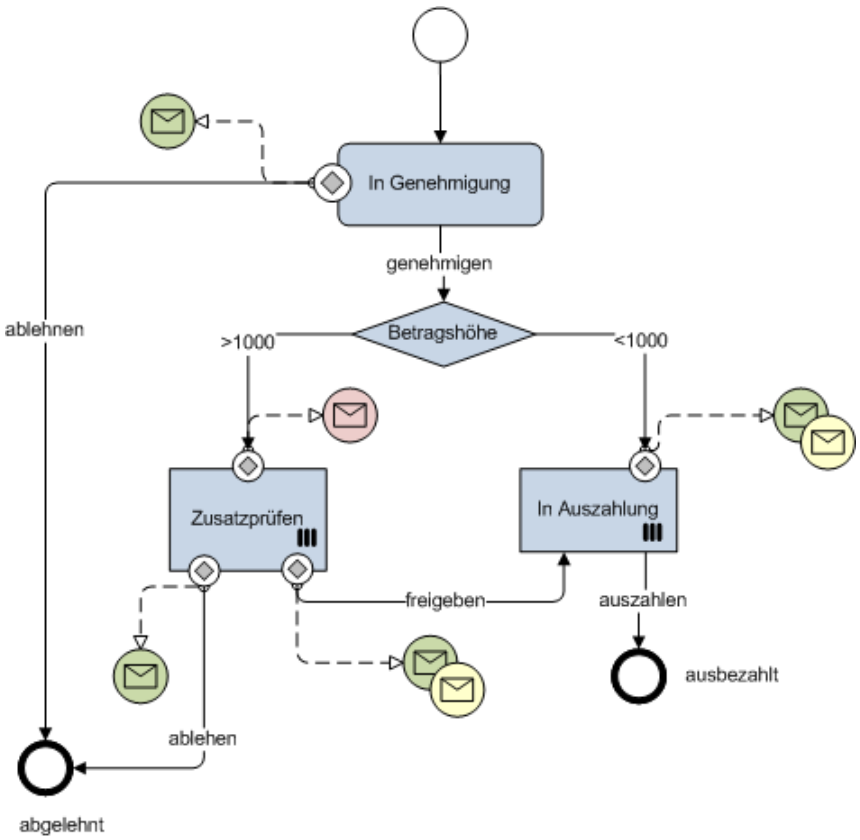
Geschäftsprozesse

Der Geschäftsprozess wird mit einem Business-Process-Diagramm dokumentiert, welches die Prozessstatus und Verknüpfungen aufzeigt. Für das Design der Prozessmodelle im Systemdesign wird der *Process Designer* der *i-engine Designer Suite* eingesetzt.

Der Prozess



Der Geschäftsprozess legt den Arbeitsfluss für das Dokument fest. Ein Prozess besteht aus einer Abfolge von seriellen und parallelen Teilaktivitäten. Zwischen den Teilaktivitäten liegt jeweils ein definierter Status des Dokumentes. Durch die Kombination von Prozesszuständen und Aktivitäten wird der Arbeitsfluss für ein Dokument festgelegt.



Einen neuen Prozess erstellen

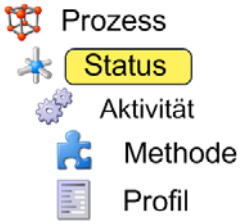
Im Ordner «Prozesse» stellt das Kontextmenü die Funktion **Prozess erzeugen** zur Verfügung. Damit wird die Eingabemaske für die Eingabe eines neuen Prozesses aufgerufen.

Feld	Beschreibung
Prozess Name	Bezeichnung für den Geschäftsprozess.
Referenz	Bezeichner; wird bei der Implementierung von Cut&Paste Operationen benötigt. s. Abschnitt «*Konfiguration Beispiel* > *Cut&Paste Operation* (Seite 259)».
Beschreibung	Prozesskurzbezeichnung (Mnemonic) und ggf. ergänzende Beschreibung. bspw. «SPA - Bearbeiten von Spesenanträgen» Der Mnemonic wird bei der Namensgebung von SQL/PL Routinen benötigt.
Typ	«PROCESS»: Normaler Prozess
Imperativ -> Vorher	Der Prozess erhält für die Anzeige drei Namen, entsprechend dem aktuellen Status (vorher – während – nachher) in dem er sich befinden kann. Hier wird der Name definiert, welcher vor dem Auslösen des Prozesses verwendet werden soll.
Präsens -> Während	Name wenn der Prozess aktiv ist.
Perfekt -> Nachher	Name wenn der Prozess beendet ist.

Hinweise

- Über das Kontextmenü des betreffenden Prozesses kann die Funktion **Prozess Bearbeiten** aufgerufen werden.
- Nicht mehr benötigte Prozesse löschen Sie über das Kontextmenü **Prozess Löschen**.
- Die Definition unterschiedlicher Prozessbezeichnungen (Vorher, Während, Nachher) macht nur Sinn, wenn der Prozess als Subprozess eingesetzt wird. Dieses Feature wird aktuell nicht verwendet. D.h. in allen drei Feldern wird der Prozessname eingefügt.

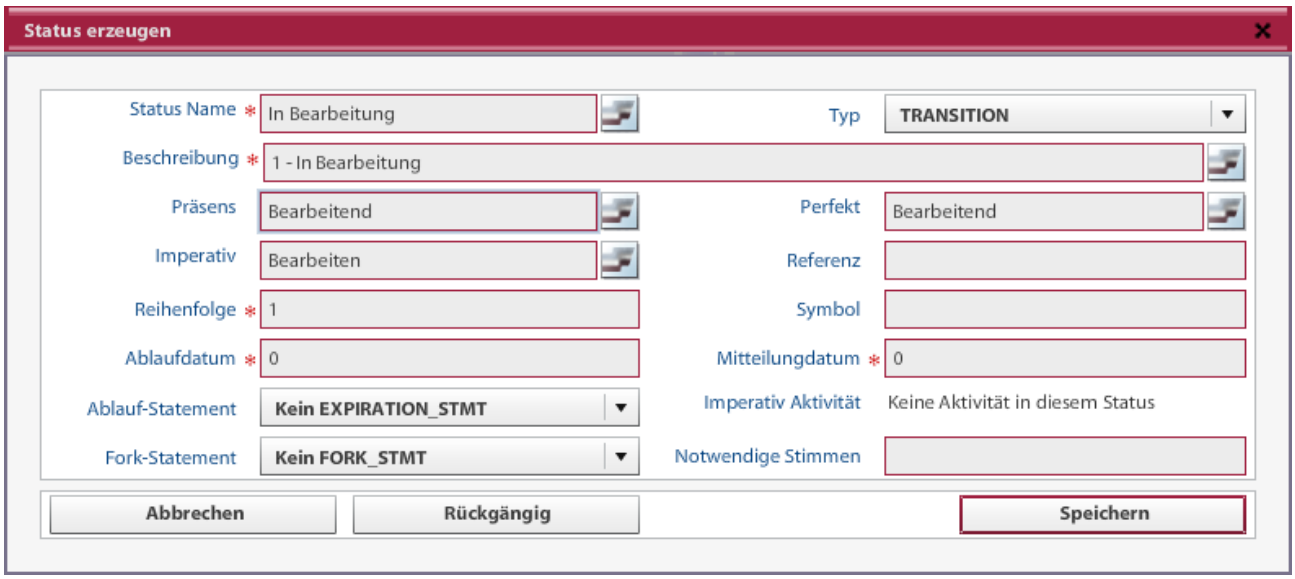
Der Prozessstatus



Während eines Geschäftsprozesses durchläuft das Dokument verschiedene Zustände z.B. «Erstellt», «Warten auf Visum», «freigegeben». Jeder vorgesehene Status wird vorgängig beim Definieren des Prozesses festgelegt.

Einen neuen Status erzeugen

Für das Erzeugen eines neuen Status steht im Kontextmenü des betreffenden Prozesses die Funktion **Status erzeugen** zur Verfügung.



Feld	Beschreibung
Status Name	Bezeichnung des Prozessstatus. Der Text sollte nicht zu lang gewählt werden, da die Überlänge sonst in der Workflow-Anzeige des Dokumentes abgeschnitten wird. Idealerweise wird die Bezeichnung aus dem Systemdesign verwendet.
Typ	«STATE»:Prozessstatus Die Berechtigung ist über die der Aktivität zugewiesenen Rollen definiert. Es werden keine dynamischen Systemrollen verwendet. «TRANSITION»: Spezielle Kennzeichnung des Zustandes während ein Dokument ausgecheckt ist. Die Berechtigung ist über die, der Aktivität zugewiesenen Rollen definiert. Es wer-

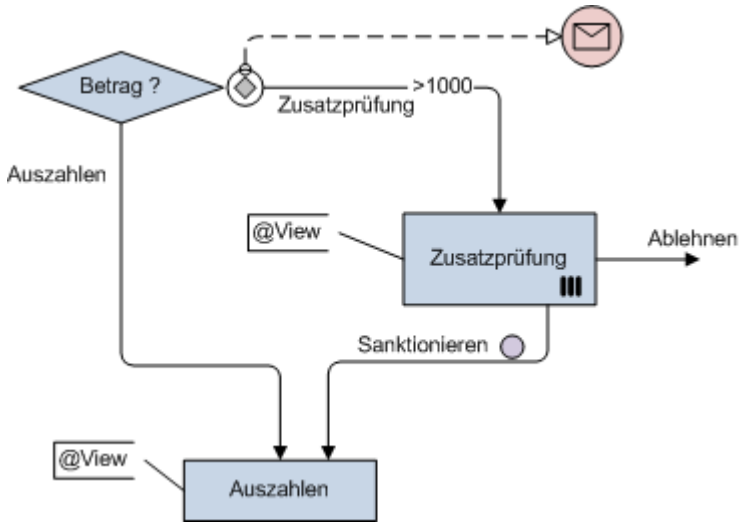
Feld	Beschreibung
	den keine dynamischen Systemrollen verwendet. «DISTRIBUTION»: Für die Berechtigung werden dynamische Rollen und Voting berücksichtigt. «SUB_DISTRIBUTION» Für die Berechtigung werden dynamische Rollen und Voting berücksichtigt. «CONDITIONAL»
Beschreibung	Prozessstatus-Kurzbezeichnung (Mnemonic) und ggf. ergänzende Beschreibung. bspw. «1 - Spesenforderung in Bearbeitung» Der Mnemonic wird bei der Namensgebung von SQL/PL Routinen benötigt.
Präsens -> Während	Statusname während der Prozess sich in diesem Status befindet.
Perfekt -> Nachher	Statusname, nachdem der Prozess diesem Status eingenommen hat.
Imperativ -> Vorher	Der Prozessstatus erhält für die Anzeige drei Namen, entsprechend dem aktuellen Status. s. Erklärungen im Abschnitt «*Statusbezeichnungen* (Seite 70)».
Referenz	Bezeichner; wird bei der Implementierung von Cut&Paste Operationen benötigt. s. Abschnitt «*Konfiguration Beispiel* > *Cut&Paste Operation* (Seite 259)».
Reihenfolge	Durchnummerierung der Prozessstatus. Diese eindeutige Nummer kann z.B. zum Referenzieren in einer Dokumentation des Prozesses verwendet werden.
Symbol	Objekte in diesem Status, werden dem Symbolbild angezeigt. Insbesondere Strukturierungsobjekte befinden sich oft in einem einfachen «Aktivierungs» Workflow. z.B. Aktiv/Passiv, ein/aus etc. Für diese Zustände werden die standardisierten Status-Symbole eingesetzt. Zusammenstellung s. Katalog - Standard Symbole. Für komplizierte Workflows müssen ggf. weitere Symbole festgelegt werden.
Verfalldatum	Maximale Anzahl Tage die der Prozess in diesem Status verbleiben soll. Anschliessend wird automatisch die Aktivität ausgelöst, welche im nächsten Feld definiert wurde. Der Wert kann auch als Dezimalzahl eingegeben werden.

Feld	Beschreibung
	Bsp. 1.75 => 42h
Ablauf-Statement	SQL-Statement das als Rückgabewert die ID der gewünschten Aktivität liefert. Das SQL-Statement ist vom Typ EXPIRATION_STMT und liefert 2 Werte: Die ID der auszulösenden Aktivität und die ID des dazu nötigen SQL Parameter Statements. Wird bei Verfall nur eine UMS-Message verschickt, bleibt dieses Feld leer.
Mitteilungsdatum	Anzahl Tage bis eine E-Mail ausgelöst wird. Der Wert kann auch als Dezimalzahl eingegeben werden
Imperative Aktivität	Wird noch nicht verwendet.
Fork-Statement	Dieses SQL-Statement wird bei jedem Protokoll-Ereignis oder regelmässig (meist 1x täglich) aufgerufen. Es überprüft – beliebige – Bedingungen und bestimmt ggf. die erforderliche Aktivität. Das SQL-Statement muss 2 Werte zurückliefern: *Aktivitäts-ID*: beliebige Aktivität, die im aktuellen Status ausgelöst werden soll. *Notiz*: Meldungstext für Protokolleintrag und Tooltip-Anzeige.
Notwendige Stimmen	Wird noch nicht verwendet.

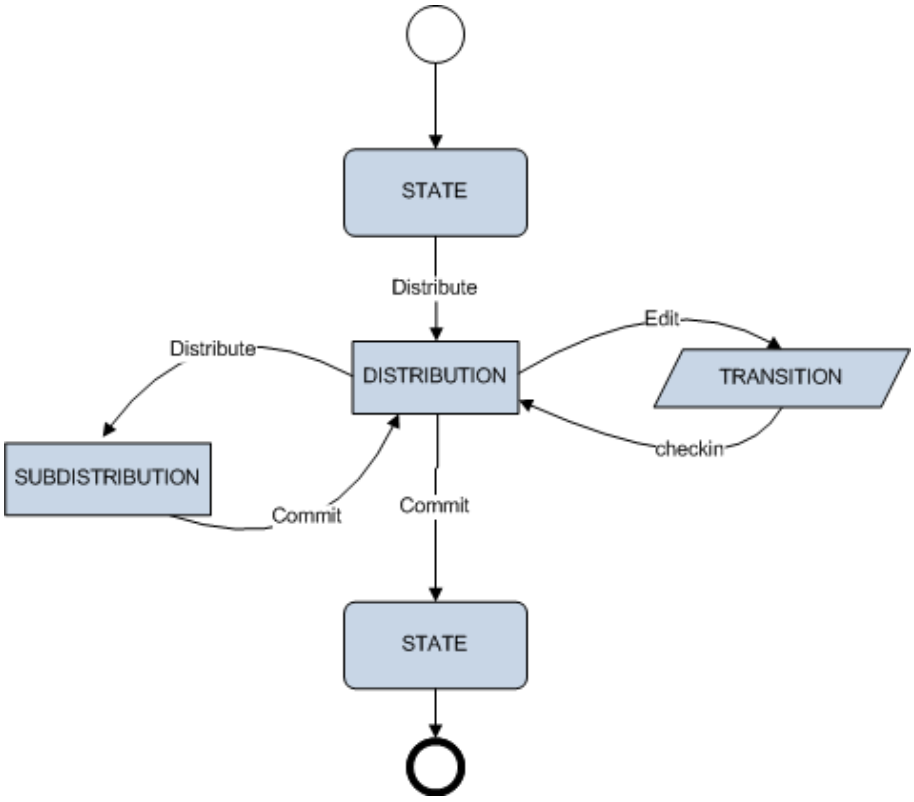
Hinweise

- Nach dem Erreichen des Ablauf- resp. Mitteilungsdatums können automatisch E-Mails versandt werden. Für die Konfiguration solche Meldungen s. Abschnitt «E-Mails im Workflow»

  Beispiel für Verwendung *Fork Statement:* Im Prozessstatus wird ein FORK-Statement ausgelöst, welches die Betragshöhe bestimmt und entsprechend die Parameter für die Distribute-Methode aufsetzt und diese auslöst.Dazu wird ein Fork-Statement aufgesetzt, das den aktuellen Betrag überprüft und entscheidet, welche der beiden Aktivitäten «Zusatzprüfung» oder «Auszahlung» der nächste Schritt im Geschäftsprozess sein wird.



- Diese Darstellung zeigt den Zusammenhang zwischen der eingesetzten Methode und der Einstellung für das Feld *Typ*.



- Der Einsatz eines Fork-Statements bedeutet immer auch die Definitionen eines entsprechenden Parameter-Statements, welches die Aufrufparameter für die eingesetzte Methode bestimmt.
- Für das Editieren muss immer ein Status des Types «TRANSITION» verwendet werden. Nur so speichert sich *i-engine* den aktuellen Benutzer, welcher das Objekt ausgecheckt hat. Nur dieser kann später das Objekt auch wieder einchecken. Im Statustyp «TRANSITION» können alle Methoden verwendet werden. Allerdings darf ein erneuter *Object.Edit()* oder eine Änderung des Objektnamens nicht durchgeführt werden. Es ist auch abzuklä-

ren, ob es sinnvoll ist, die Attribute eines ausgecheckten Objektes durch andere Benutzer/Anwendungen ändern zu lassen.

Katalog - Standard Symbole für Prozessstatus

Statusbezeichnungen

Für den Prozessstatus müssen mehrere Bezeichnungen festgelegt werden. Mit unterschiedlichen Bezeichnungen lassen sich unterschiedliche Zeitpunkte im Ablauf unterscheiden. Obwohl nicht zwingend, empfiehlt es sich jeweils die richtigen Zeitformen zu verwenden.

Feld	Erklärung	Beispiele
Status Name	Bezeichnung des Status (resp. der Objekte, welche sich in diesem Status befinden).	Prüfung Bearbeitung Archivierung
Imperativ -> Vorher	Handlungsaufforderung an den Benutzer (z.B. beim Öffnen des Aufgabenordners.	Prüfen Sie Bearbeiten Sie Archivieren Sie
Präsens -> Während	Wird als Tooltip im Listenfeld verwendet.	Prüfen Bearbeiten

Feld	Erklärung	Beispiele
		Archivieren
Perfekt -> Nachher	Bezeichnung nach Verlassen des Status.	Geprüft (resp. Abgelehnt) Bearbeitet Archiviert
Beschreibung	Ausführlichere Beschreibung Dieses Feld ist zur Dokumentation und wird nicht ausgewertet.	Prüfung des Dokuments durch Stelle, Q-Prüfung nach erfolgter Freigabe

Zwischenstatus

Der Zwischenstatus wird verwendet, um die gleichzeitige Verwendung eines Objektes durch mehrere Benutzer zu verhindern. Dazu wird ein Objekte, welches sich in Bearbeitung befindet, in den Zwischenstatus gesetzt und erst nach der Abarbeitung wieder in den ursprünglichen Status gesetzt. Dadurch können andere Benutzer das gleiche Objekt nicht ebenfalls bearbeiten.

Im Fehlerfall wird das Objekt automatisch in den ursprünglichen Status zurückgesetzt. Dazu kontrolliert ein Keep-Alive Mechanismus folgende Bedingungen:

- Der Client meldet sich über längere Zeit nicht mehr (Timeout)
- Der Client führt die Aktivität nicht bis zum Schluss aus (Died)

Im Zwischenstatus muss die Methode *Object.RestoreStatus* parametrisiert werden, welche das Objekt wieder in den ursprünglichen Status zurückführt.

DB Action

Die Verwendung des Zwischenstatus wird durch Einträge in der Tabelle OBJECT_PROPERTY aufgesetzt:

OBJ_ID	Activity-ID
SEQ	«0»
KEY / VALUE	«INTERMEDIATE_STATUS_ID» Entweder wird die Status-ID des Zwischenstatus angegeben oder der Wert «DEFAULT». In diesem Fall wird der Standard Status 10080769 verwendet, bei welchem bereits die Methode *Object.RestoreStatus()* parametrisiert ist.

Wie es funktioniert

Beim Start der Aktivität wird geprüft ob ein Zwischenstatus parametrisiert ist und das Objekt ggf. in den entsprechenden Status gesetzt. Anschliessend wird ein KeepAliveThread für die Aktivität gestartet. Dieser Thread prüft periodisch, ob der Client noch aktiv ist und ruft den KeepAliveCallbackHandler (die Methode) auf, falls dies nicht der Fall ist. Ebenso wird der CallbackHandler aufgerufen, falls ein «died» oder «done» Ereignis eintrifft.

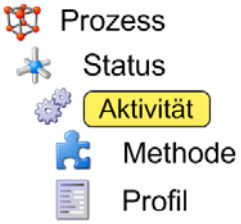
Der HTMLClient schickt AJAX Requests an das KeepAliveServlet um den Keep Alive aufrecht zu erhalten. Ebenfalls meldet der Client die Ereignisse «died» und «done». Beide Ereignisse eliminieren den KeepAliveThread. Bei Win Client und Flex Client ist keine KeepAlive Mechanismus implementiert, so dass der Zwischenstatus nach 2 Minuten automatisch beendet wird..

Wird die Aktivität irregulär beendet («timeout» oder «died»), so wird die Methode das Objekt wieder in den ursprünglichen Status zurücksetzen. Wird die Aktivität regulär beendet («done»), so wird der Workflow Manager das Objekt wieder in den ursprünglichen Status zurücksetzen.

Hinweis

- Dieser automatisierte Zwischenstatus kann nicht bei der Methode *Object.Edit()* eingesetzt werden.

Die Aktivität



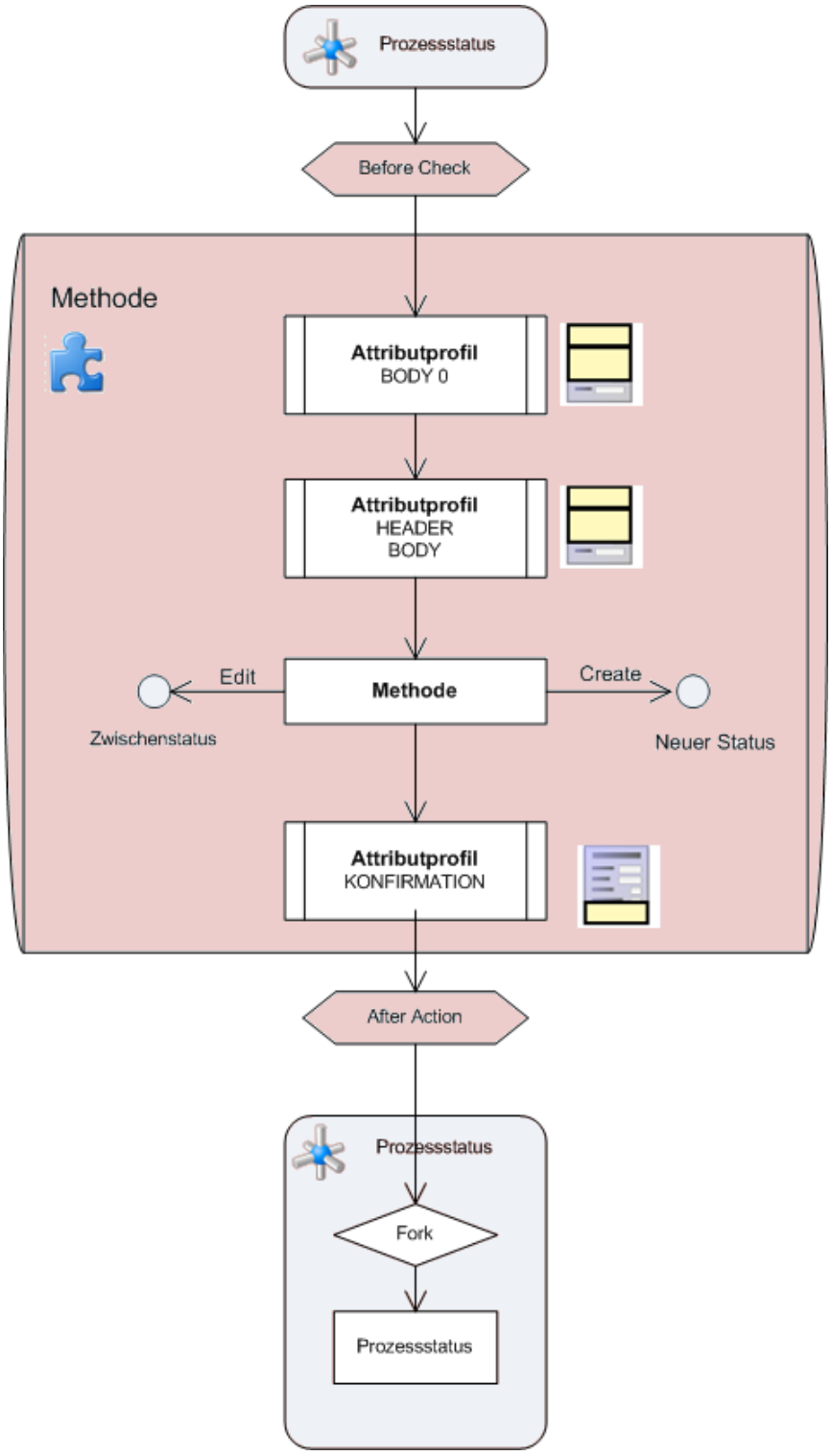
Abhängig vom Status eines Dokumentes sind weitere Arbeitsschritte notwendig resp. möglich. Diese Aktivitäten (Bearbeitungen, Arbeitsprozesse oder Funktionen) sind Einzelschritte des gesamten Geschäftsprozesses, z.B. «Modifizieren», «Visum erteilen», «Verteilen». Eine Aktivitäten wird vom einem Bediener angestossen oder im Verlauf des Geschäftsprozesses automatisch ausgelöst.

Nachdem die Aktivität ausgeführt wurde, hat das Dokument wieder einen durch den Workflow definierten Status eingenommen.

Welche Aktivitäten in welchem Prozesssstatus erforderlich sind, wird im Rahmen des Systemdesigns vom Prozessverantwortlichen definiert.

Genereller Ablauf einer Aktivität

Die Aktivität wird einerseits durch den Bediener angestossen indem er eine bestimmte Funktion in der Menüleiste, dem Kontextmenü oder über Anklicken des Eintrages im Struktur- oder Datenfenster auslöst. Andererseits kann eine Aktivität auch durch Ereignisse im Workflow automatisch ausgelöst werden.



Before Checker

Nach der manuellen oder ereignisgesteuerten Auslösung überprüft der «Before Check», ob unter den aktuellen Voraussetzungen die Aktivität überhaupt ausgeführt werden darf/kann. Er verwendet dazu ein entsprechendes SQL-Statement, welches die Freigabe als Ergebnis zurückliefert.

Anzeige Attributprofil

Während der Abarbeitung der Methode werden ggf. mehrere Masken angezeigt. Falls für die Methode eine Eingabe notwendig wird – z.B. *Object.Edit()*– wird als erstes eine Eingabemaske entsprechend den Definitionen des zugeordneten Attributprofils (Anwendungsbereich «BODY») angezeigt. Dieser Maske kann auch eine Auswahlmaske für eine Vorlage vorgeschaltet werden (Anwendungsbereich «BODY0»).

Die Methode wird dann zu Ende ausgeführt. Beim HTML Client wird zur Bestätigung immer eine Maske entsprechend den Definitionen des Attributprofils (Anwendungsbereich «CONFIRMATION») angezeigt. Deshalb muss jeder Aktivität ein minimales Attributprofil mitgegeben werden. Bei den anderen Clients erfolgt die Rückmeldung über die Statuszeile - aus Kompatibilitätsgründen sollte das Attributprofil aber immer einen «CONFIRMATION» Teil enthalten.

After Action

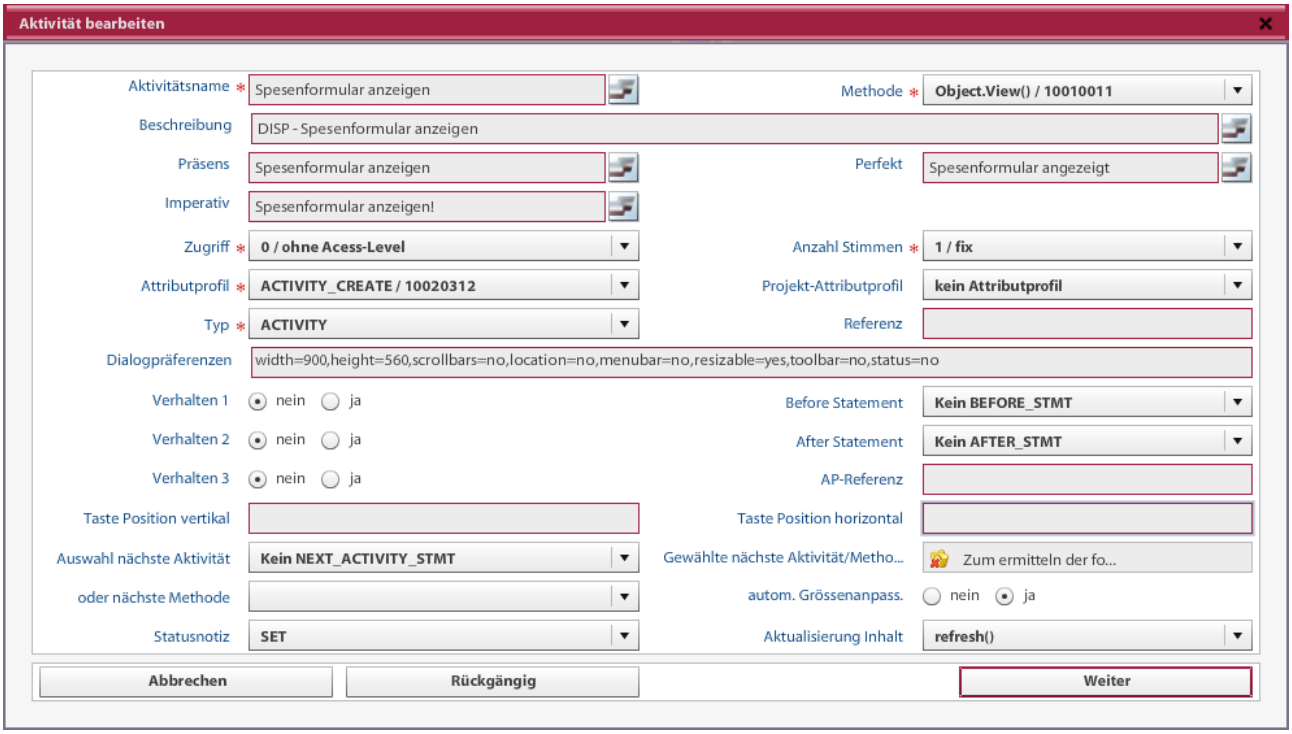
Als letzte Funktion kann die Aktivität ein sog. «After Action» durchführen, die Daten in der Datenbank nachführen kann; oder eine weitere Aktivität aufrufen. Bei neuen Implementationen ist auf den Einsatz dieses Features zu verzichten.

Fork-Statement

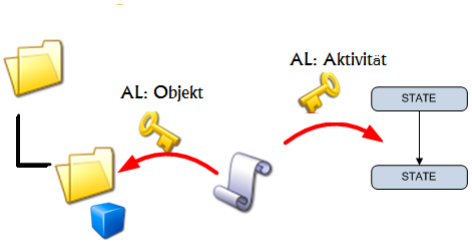
In der Aktivität kann auch ein «Fork-Statement» festgelegt werden. Dieses ist aber bereits Teil des nächsten Prozessstatus im Workflow.

Eigenschaften

Für das Erzeugen oder Ändern einer Aktivität steht im Kontextmenü des betreffenden Prozessstatus die Funktion **Aktivität erzeugen** resp. **Aktivität bearbeiten** zur Verfügung.



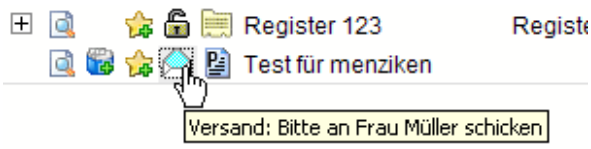
Feld	Beschreibung
Aktivitätsname	Bezeichnung der Aktivität. Der Text sollte nicht zu lang gewählt werden, da er als Auswahltext im Kontextmenü erscheint. Bei Eingabe kontrollieren, dass die Einträge für die benötigten Sprachvarianten aufgesetzt sind. Idealerweise wird die Bezeichnung aus dem Systemdesign übernommen.
Methode	*i-engine* stellt eine Reihe von Methoden zur Verfügung. Siehe *Referenzhandbuch*.
Beschreibung	Ergänzende Beschreibung; erscheint nur in der Objektlisten-Anzeige der Aktivitäten. Ist damit für den Endanwender nicht relevant, dient als Zusatzinformation für den Parametriker. In den Bezeichner für SQL/PL Routinen wird oft ein Teil der Methodenbezeichnung eingebaut. Diese wird der Aktivitäts- Kurzbezeichnung (Mnemonic) wird der *Beschreibung* vorangestellt. bspw. «DISP - Spesen-

Feld	Beschreibung
	formular anzeigen»
Präsens	Wird für die Anzeige in der Statuszeile verwendet.
Perfekt	Das Ausführen der Aktion wird protokolliert. Für den Eintrag wird diese Bezeichnung verwendet
Imperativ	Tooltip Text, welcher beim entsprechenden Aktivitäten-Symbol in der Objektliste angezeigt wird.
Zugriff	Die Ausführungsberechtigung für die Aktivität kann über Berechtigungsstufen vergeben werden. Im Feld *Zugriff* wird der minimal erforderliche Level festgelegt. Folgende Berechtigungsstufen sind vorgegeben: «0»: nur durch Rolle bestimmt «5»: Verzeichnis anzeigen «10»: lesen «30»: lesen/schreiben «50»: delegierter Besitzer «70»: Besitzer  Die AL-Objekt muss tiefer/gleich der AL-Aktivität sein. Andernfalls kann die Aktivität gar nicht ausgelöst werden, da ja das Objekt nicht im Zugriff ist.
Anzahl Stimmen	Mit dem Ausführen der Aktivität wird der Prozess gemäss dem festgelegten Workflow aus dem «Aktuellen Status» in den «Nächsten Status» überführt. Mit dem Wert von *Anzahl Stimmen* kann festgelegt werden, wie oft die Aktivität ausgelöst werden muss, bis der nächste Prozessstatus eingenommen werden soll. Üblicherweise wird dieser Wert auf 1 gesetzt, da meist eine einmalige Ausführung genügt. Das Feld kann aber auf jeden beliebigen Wert >«0» gesetzt werden. Beispiel: Durch eine Aktivität werden Kursanmeldungen erfasst. Nach max. 10 Anmeldungen ist der Kursausgebucht. Für die Aktivität "Kurs ausgebucht"

Feld	Beschreibung
	wird *Anzahl Stimmen* auf «1/10» gesetzt. Für die Methode *Object.Commit* gilt folgende Ausnahme: Es können folgende Werte spezifiziert werden: «0»: Die Stimme eines Teilnehmers bei einem «Commit» hat 1/n Wert (n = Anzahl Teilnehmer) «1»: Die Stimme des Teilnehmers hat 1/1 Wert. d.h. die Vernehmlassung wird beendet.
Anzeige (Hauptfenster) (Projektfenster)	Für die Anzeige verwendet die Methode die zugeordnete Attributliste. Legen Sie fest, in welchem Fenster die Anzeige erfolgen soll. Es können unterschiedliche Anzeigeoptionen gewählt werden, abhängig davon, ob die Aktivität im Haupt- oder einem Projektfenster aktiviert wird. «_blank»: in einem neuen Fenster «content»: im aktuellen Fenster im Listenfeld. «executor»: kein Fenster, es wird nur Statustext (Beschreibung der Aktivität) in der Statuszeile angezeigt. Geeignet zur Benutzerorientierung beim direkten Ausführen von Abläufen. «_top»: Anzeige im Hauptfenster
Attributprofil	Für die Anzeige und ggf. Erfassung wird eine Dialogmaske via Attributprofil zugewiesen. Die gleiche Attributsliste wird auch für die Ausgabe der Bestätigungsmeldung nach dem Ausführen der Methode verwendet.
Projekt-Attributprofil	Das Projekt-Attributprofil wird nur im Zusammenhang mit der Methode *Object.Paste()* verwendet, wenn vorgängig *Project.Cut()* resp. *Project.Copy()* ausgeführt wurde. s. Abschnitt «*Cut&Paste Operationen > Projekt* (Seite 266)»
Typ	Durch die Wahl des Typs wird das übergeordnete Verhalten der eingesetzten Methode definiert. Es werden nur die beiden folgenden Werte verwendet: «ACTIVITY»: Die Methode wird als «normale» Aktivität verwendet. Dies ist der übliche Fall. «CONSTRUCTOR»: Nur für Create-Methoden: Das Objekt, worauf die Aktivität ausgeübt wird, wird als Vorlage für ein neues Objekt verwendet. Z.B. *Object.Create()* oder *Tree.Clone()*.

Feld	Beschreibung
	«TRANSITION»
Referenz	Einige Methoden (z.B. *project.Update*) können eine Aktivität in untergeordneten Client-Objekten auslösen. Dabei können sich die Client-Objekte in ein einem Prozessstatus befinden, welcher mehrere Aktivitäten zulässt. Die gewünschte Aktivität muss deshalb über eine Zusatzinformation bestimmt werden. Das Feld *Referenz* stellt den Zusammenhang zwischen Aktivität im Parent-Objekt und Aktivität in den Client-Objekten her. In den Clients wird jene Aktivität ausgelöst, welche den gleichen Referenz-String wie die Parent-Aktivität hat.
Dialogpräferenzen	Falls die Anzeige in einem neuen Fenster erfolgt, kann die gewünschte Fenstergrösse gewählt werden: Die Eingabe entspricht der normalen HTML-Syntax. Beispiel: «width=770, height=520, scrollbars=no, location=no, menubar=no, resizeable=yes, toolbar=no, status=no» Hinweis: Das Feld sollte nicht leer sein. ggf. «NONE» einfügen. Ausser beim Einsatz für Registerfenster wird üblicherweise die automatische Grössenanpassung eingesetzt.
Verhalten	Dieser Parameter regelt das Verhalten der Objekte im Strukturfenster. Das Verhalten kann unabhängig für das Hauptfenster, das Projektfenster und das Popup festgelegt werden. Wenn das Objekt in der linken Baumanzeige dargestellt wird, kann mit einem Mausklick auf das Objekt eine Aktivität ausgelöst werden. Abhängig vom aktuellen Status des Objektes stehen die zugeordneten Aktivitäten zur Auswahl. Eine – und nur eine - dieser Aktivität kann zur Ausführung gekennzeichnet werden. «JA»: Aktivität beim Anklicken verwenden «NEIN»: Aktivität nicht verwenden Hinweis: In jedem Status sollte nur bei einer einzigen Aktivität das Verhalten auf «JA» gesetzt sein. Andernfalls erscheint das Objekt mehrfach im Baum.
Before Statement	Das *Before-Statement* kann vor der Ausführung der Methode die Voraussetzungen oder Datenbestände

Feld	Beschreibung
	überprüfen. Je nach Ergebnis kann die Ausführung dann gestoppt oder doch durchgeführt werden. Als *Before-Statement* wird ein SQL-Statement eingesetzt, welches die notwendigen Rückgabewerte liefert. s. Abschnitt zu «*SQL-Statements – Before-Statement* (Seite 40)»
After Statement	Über das After Statement können nach Abschluss der Methode noch Schreiboperationen in der Datenbank durchgeführt werden. Beispiel: die Methode hat einen Master-Record nachgeführt. In der Folge müssen alle Client-Records ebenfalls nachgeführt werden. Hinweis: Der Einsatz von *After Statements* ist nur in Ausnahmefällen vorgesehen.
AP-Reference	s. Erläuterungen im folgenden Abschnitt «*AP-Reference*».
Tasten Position vertikal / horizontal	Das Projektfenster stellt eine eigene Menüleiste zur Verfügung. Durch das Festlegen der Button-Position kann für die Aktivität eine Schaltfläche darin generiert werden. Die Positionswerte starten ab «1». Der Wert «0» bedeutet, dass keine Schaltfläche angezeigt wird. Als Schaltflächentext wird der Aktivitätsname genommen.
Nächste Aktivität oder Nächste Methode	Nach Beendigung der aktuellen Aktivität kann ohne weitere Interaktion eine weitere Aktivität gestartet werden. Die Object-ID dieser Aktivität wird als Wertresultat von einem SQL-Statement zurückgemeldet. Meist ist die nächste Aktivität fix zugewiesen, so dass im SQL-Statement ein «SELECT 4711 FROM DUAL» hinterlegt werden kann. Hinweis: die nächste Aktivität muss das gleiche *Target* wie die aktuelle Aktivität verwenden. Wenn durch die aktuelle Methode ein neues Objekt erzeugt wird, wird die «Nächste Aktivität» immer auf dieses neue Objekt angewandt. Die betreffende Aktivität muss im Prozessstatus, welches das Objekt nach Ausführen der aktuellen Methode einnehmen wird, definiert sein.

Feld	Beschreibung
	Bei Methoden *Create* oder *Import* wird oft eine darauffolgende *Edit*-Methode eingesetzt. In diesem Fall kann auf die Erstellung eines SQL-Statements verzichtet und direkt die *Nächste Methode* festgelegt werden. **Wichtig: Die Angabe einer Methode ist die bevorzugte Variante. Damit entfällt das Definieren eines SQL-Statements und es wird keine applikationsspezifische ID benötigt.**
Gewählte nächste Aktivität/Methode	Resultatfeld für die Auswahl aus *Nächste Aktivität* und *Nächste Methode.*
Autom.: Grössenanpassung	Die Fensterhöhe für den Dialog kann automatisch berechnet werden. Trotzdem empfiehlt es sich, die *Dialogpräferenzen* korrekt aufzusetzen – dadurch wird der Flackereffekt nach dem ersten Anzeigen des Fenster vermieden. Ausnahme sind Dialoge bei denen Register eingesetzt werden – z.B. Eigenschaften, Attributprofile, Projektfenster. Da der Platzbedarf der Folgeregister grösser sein kann als jener des ersten, müsste für die Bedienung jeweils gescrollt werden. Ebenfalls keine autom. Grössenanpassung kann bei *Create*-Methoden mit Vorlagenauswahl verwendet werden. «JA»: Grösse automatisch anpassen «NEIN»: Einstellung aus Dialogpräferenzen verwenden Hinweis: Die automatische Grössenanpassung wird nicht von allen Methoden unterstützt.
Statusnotiz	In der Listenanzeige kann auf dem Prozessstatus-Symbol ein Tooltip angezeigt werden. Der angezeigte Text stammt aus dem WF_NOTE Feld des Attributprofils.  Im gezeigten Beispiel stammt: «Versand» aus Prozessstatus Präsenstext «Bitte an Frau...» aus WF_NOTE Feld Über die *Statusnotiz* kann festgelegt werden, unter welchen Bedingungen der Tooltip Inhalt verändert

Feld	Beschreibung
	werden soll. «SET»: Der Tooltip erhält den Text, welcher bei dieser Aktivität in das WF_NOTE Feld eingegeben wurde. Wurde das Feld nicht angezeigt oder kein Text eingegeben erscheint später kein Tooltip. «CLEAR»: Der Tooltip wird gelöscht «KEEP»: Der aktuelle Inhalt des Tooltip wird nicht verändert. Unabhängig von einer eventuellen Eingabe in WF_NOTE bleibt der bisherige Tooltip erhalten. Generell wird der Tooltip Inhalt nur verändert, wenn die Aktivität einen *Nächsten Status* definiert hat (kann auch der gleiche Status sein).
Aktualisierung Inhalt	Nach einer Aktivität kann es notwendig werden Teile oder den ganzen Anzeigebereich nachzuführen. Es sind verschiedene Refresh-Scripts verfügbar, so dass gezielt die betroffenen Elemente aktualisiert werden können. *Festlegen des Fensters:* «Mainframe»*:* Hauptfenster «Project»: Projektfenster «Nearest»: die Liste der aufrufenden Fenster wird zurückverfolgt, bis das gewünschte Frame gefunden wird. *Festlegen des Fensterbereichs:* «Tree»: Strukturfenster «List»: Datenfenster «Inner»: Struktur- und Datenfenster Beispiel: «refreshMainframeInner()» -> beide Frames des Hauptfensters «refreshProjectTree()» -> Strukturanzeige im Projektfenster «refreshNearestTree()»

Feld	Beschreibung
	-> Struktur im aufrufenden Fenster. Spezialfall: «refreshProject()» lädt mit der Methode *project.edit()* das gesamte Projekt neu, so dass auch nach einem Statuswechsel auf dem Projekt weitergearbeitet werden kann. «refresh()»: entspricht einem «refreshNearestInner()» und wird meist verwendet.

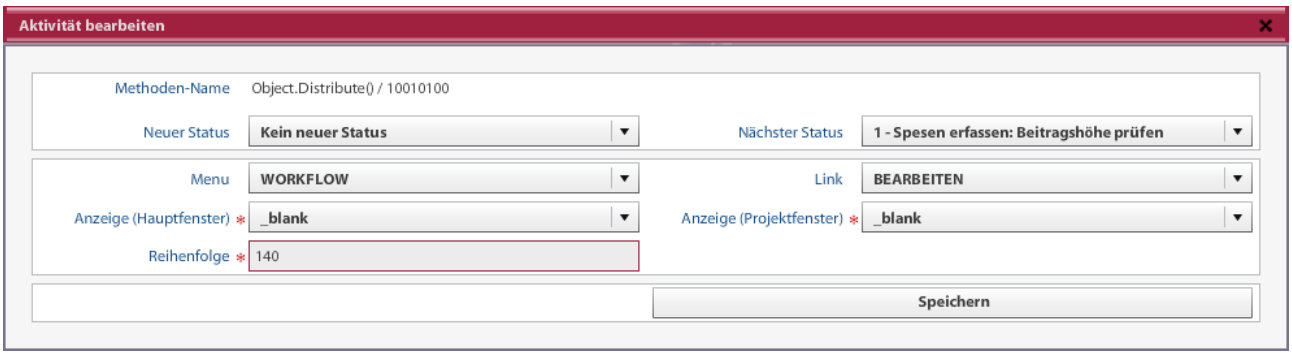
Hinweise

- Es können mehrere Aktivitäten mit gleichem Aktivitätsname und Methode definiert werden. Anwendungsfall ist eine Funktion, die von Benutzern mit einer bestimmten Berechtigungsstufe oder einer bestimmten Rollenmitgliedschaft ausgelöst werden soll. Es sollte aber sichergestellt sein, dass der Benutzer die Rechte nur jeweils auf eine Aktivität gleichzeitig hat. Andernfalls wird ihm jene mit der tiefsten Sequenznummer angezeigt.
  Beispiel: Ein Systemadministrator hat via Berechtigungsstufe keinerlei Rechte, um auf Dokumente der Benutzergruppe irgendwelche Aktivitäten auslösen zu können. Er kann somit auch keine Dokument **Einchecken**. Um aber dem Anwender bei Problemen Hilfestellung geben zu können, ist die Aktivität **Einchecken** ein weiteres Mal definiert und der Administratorrolle wird die entsprechende Berechtigung erteilt. Jeder Benutzer sieht dabei nur entweder die eine oder die andere Aktivität.
- Die Reihenfolge der Berechtigungsstufen ist später praktisch nicht erweiterbar. Deshalb sollten Lücken für spätere Ergänzungen vorgesehen werden. Neue Berechtigungsstufen müssen direkt in der Datenbanktabelle P_CODE mit TAB_NAME=«POPUP_ACCESS» eingerichtet werden.
- Es darf immer nur eine einzige Aktivität im Feld *Verhalten* mit «JA» markiert werden. Als Vorgabe wird jeweils die erste definierte Aktivität entsprechend markiert.
- Verhalten und Berechtigung: Es kommt zu Fehlermeldungen, wenn ein Objekt zwar im Baum angezeigt werden kann, aber das Ausführungsrecht für jene Aktivität, die per *Verhalten*-Einstellung definiert wurde, nicht erteilt wurde.
- Die Informationen in den Feldern *Präsens*, *Perfekt* und *Imperativ*: Typischerweise wird in *Präsens* und *Imperativ* genau der *Aktivitätsname* aufgeführt. In Ausnahmefällen kann des sinnvoll sein, im *Präsens* einen Text einzugeben zum Andauern der Operation einzugeben. da dieser in einem Meldungsfenster speziell angezeigt wird. Der Text in *Perfekt* sollte eine sinnvolle Protokollierung erlauben.
- Unabhängig vom Wert in *Statusnotiz* wird im Protokoll immer der eingegebene Text aus WF_NOTE aufgeführt.

- Die Möglichkeit Stimmen anteilig zu verwenden, wird praktisch nur bei *Distribute* Methoden verwendet. Andere Anwendungsfälle lassen sich besser über Fork-Statements realisieren.

Methodenmaske

Über die Schaltfläche **Weiter** wird die Methodenmaske geöffnet.



Der Inhalt dieser Maske ist von der verwendeten Methode abhängig. Die entsprechenden Hinweise zu jeder Methode finden sich im *Referenzhandbuch*. Eine generelle Beschreibung finden Sie im Abschnitt «*Die Methode* (Seite 99)».

Feld	Beschreibung
Neuer Status	Bezeichnung der Aktivität. Der Text sollte nicht zu lang gewählt werden, da er als Auswahltext im Kontextmenü erscheint. Bei Eingabe kontrollieren, dass die Einträge für die benötigten Sprachvarianten aufgesetzt sind.
Nächster Status	
Menu	
Link	
Anzeige (Hauptfenster)	
Anzeige (Projektfenster)	

Weitere Einstellungen

DB Action

Abhängig von der eingesetzten Methode können noch weitere Einstellungen in folgenden Tabellen vorgenommen werden.

- OBJECT_PROPERTY s. Abschnitt «*Projektfenster* (Seite 197)»
- WF_DECISION s. Abschnitt «*Cut&Paste Operationen* (Seite 259)»
- WF_ACTIVITY_PROFILE s. folgender Abschnitt «AP-Reference»

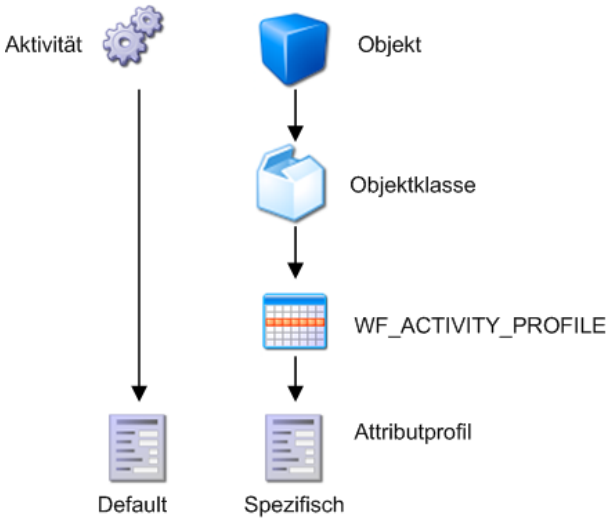
Die entsprechenden Details sind auch in der «Methoden Referenz» aufgeführt.

AP-Reference

Aus Optimierungsgründen sollte ein Prozess für eine möglichst umfassende Gruppe von Objektklassen eingesetzt werden können. Häufig sind aber für einige der Objektklassen in einer solchen Gruppe z.T. unterschiedliche Attributprofile notwendig.

Wenige Abweichungen

Falls es sich um eine überschaubare Anzahl Objektklassen handelt, für welche ein abweichendes Attributprofil notwendig wird, wird folgendes Verfahren verwendet.



1. In der Aktivität wird im Feld *Attributprofil* jenes Attributprofil definiert, welches meist verwendet werden soll.

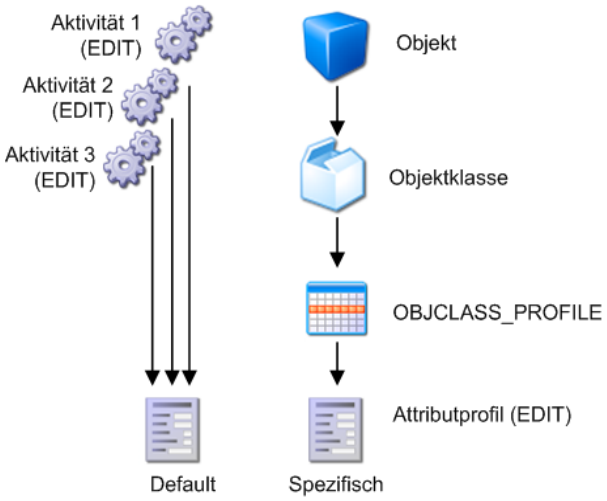
2. Alle Abweichungen werden in Tabelle WF_ACTIVITY_PROFILE eingetragen.

ACTIVITY_ID	OBJCLASS_ID	ATTRIBUTE_PROFILE_ID
10011072489	10011071250	10011071473
10011072489	10011071257	10011071473
10011072489	10011071265	10011071473
10011072489	10011071273	10011071473
10011072489	10011071280	10011071473
10011072489	10011071287	10011071473

Für jede Objektklasse wird ein Eintrag mit dem zu verwendenden Attributprofil erstellt. Diese Einträge sind natürlich ggf. für alle betroffenen Aktivitäten zu erstellen.

Viele Abweichungen (in Arbeit)

Falls es notwendig wird, eine grosse Anzahl Objektklassen zu unterhalten – z.B. da jede ein eigenes Symbol besitzen soll – ist die vorgehende Lösung sehr mühsam. Mit der folgenden Methode muss die die Zuweisung Objektklasse zu Profil nur 1x pro Profiltyp und nicht pro Aktivität gemacht werden.



1. In der Aktivität wird im Feld *Attributprofil* jenes Attributprofil definiert, welches meist verwendet werden soll. Zusätzlich wird im Feld AP_REFERENCE ein Bezeichner für die Gruppe der zu verwendenden Attributprofile aufgesetzt.

2. In der Tabelle OBJCLASS_PROFILE wird die Zuordnung zwischen Attributprofil und Bezeichner für jede Objektklasse festgehalten.

OBJCLASS_ID	ATTRIBUTE_PROFILE_ID	AP_REFERENCE
10011070008	10011069313	EDIT
10011070011	10011069313	EDIT
10011070014	10011069313	EDIT
10011070017	10011069313	EDIT
10011070020	10011069313	EDIT
10011070023	10011069313	EDIT

3. Aktuell muss mit dieser Information nun noch die Tabelle WF_ACTFITY_PROFILE nachgetragen werden. Dies übernimmt ein Script.

Hinweis

Es ist üblich, dass pro Objektklasse 6 Attributprofile notwendig werden: Edit, View, Create, Version, Import und Paste. Da diese Attributprofiltypen in mehreren Aktivitäten wiederverwendet werden, bietet der «AP Reference» Ansatz hier eine Optimierung. In der OBJCLASS_PROFILE Tabelle muss nur 1x pro Profiltyp und nicht wie in der WF_ACTIVITY_PROFILE Tabelle für jede Aktivität und Profil ein Eintrag erfolgen muss.

Nomenklatur

Um dem Benutzer eine möglichst eingängige Bedienung über die Kontextmenüs zu bieten, sollten folgende Richtlinien berücksichtigt werden.

Aktivität	Symbol	Kontext Sequenz	Methoden
Erzeugen			
Aktivität erzeugen		B600	Activity.create
Rollen zuteilen		B 615	Activity.AddRole
Attributprofil erzeugen		B600	AttributeProfile.Create
Erweiterung erzeugen		B525	Extension.Create
Sucher erzeugen Filter erzeugen Bericht erzeugen		B630	Finder.Create
iTan Codes erzeugen			iTan.Create
Ablage erzeugen		B650	Location.Create
Objektklasse erzeugen		B510	Objclass.Create
Referenz einfügen		E175	Object.AddAlias
Verknüpfung erstellen		E165	Object.AddLink
Objekt erzeugen		B45	Object.Create
Referenz erzeugen		E170	Object.CreateAlias
Verknüpfung erzeugen		E160	Object.NewLink
Abo setzen		E	Object.SetSubscription
OU erzeugen		B615	OU.Create
Patient erzeugen			Patient.Create
Prozess erzeugen		B285	Process.Create
Benutzer hinzufügen Benutzer anzeigen		B430	Role.AddUser
Rolle erzeugen		B425	Role.Create
Service erzeugen			Service.Create

Status erzeugen		B555	Status.Create
Stylesheet erzeugen		B585	Stylesheet.Create
Duplizieren		E135	Object.Clone
Erzeugen		E65	Tree.clone
Rolle hinzufügen		B480	User.AddRole
Stellvertreter setzen		E485	User.AddProxy
Benutzer erzeugen		E450	User.Create

Aktivität	Symbol	Kontext Se-quenz	Methoden
Löschen			
Löschen		B610	Activity.Delete
Löschen		E625	Alias.Delete
Löschen		B580	Attributprofil.Delete
Leeren			Cart.RemoveAll
Löschen		E545	Clone.Delete
Löschen		B535	Extension.Delete
Löschen		B645	Finder.Delete
Löschen			iTan.Delete
Löschen		B660	Location.Delete
Löschen		B520	Objclass.Delete
Permanent löschen Löschen		E095	Object.Delete
Löschen		E 245	Object.RemoveLink
Archivkopie löschen		E	Object.

			RemoveFromLocation
Löschen		E095	Object.Scrap
Löschen		B295	Process.Delete
Löschen		E 355	Project.Delete
Referenz löschen		E15	Project.DeleteAlias
Dossier löschen			Project. RemoveFromLocation
Referenz löschen		B400	Project.RemoveLink
Löschen		B440	Role.Delete
Löschen		E	Service.Delete
Löschen		B565	Status.Delete
Löschen		B595	Stylesheet.Delete
Löschen		E 505	User.Delete

Aktivität	Symbol	Kontext Sequenz	Methoden
Bearbeiten			
Bearbeiten		B605	Activity.Edit
Bearbeiten		B575	Attributprofil.Edit
Bearbeiten		B530	Extension.Edit
Bearbeiten		B640	Finder.Edit
Versionsnummer bear-beiten		B665	Index.Edit
Bearbeiten			Instance.Edit
Bearbeiten			iTan.Edit
Bearbeiten		B655	Location.Edit

Bearbeiten		B515	Objclass.Edit
Eigenschaften bearbeiten		B20	Object.Attributes.Edit Project.Attributes.Edit
Gruppe bearbeiten		E	Project.BulkAttributesEdit
Bearbeiten		B30	Object.Edit
Bearbeiten			OU.Edit
Bearbeiten			Patient.Edit
Bearbeiten		B655	Process.Edit
Dossier bearbeiten		B8	Project.Edit
Bearbeiten		B435	Role.Edit
Bearbeiten			Series.Edit
Bearbeiten		E	Service.Edit
Bearbeiten		B560	Status.Edit
Bearbeiten		B590	Stylesheet.Edit
Bearbeiten			Study.Edit
Eigenschaften bearbeiten		B460	User.Properties.Edit
Bearbeiten			Visit.Edit

Aktivität	Symbol	Kontext Sequenz	Methoden
Referenz aktualisieren		E25	Alias.Move2MaxVersion
Duplikate synchronisie-ren		B615	Clone.Synchronize
Aufgaben anzeigen Verlauf anzeigen			Constraint.ObjList
Anzeigen		E635	Finder.Report
Archivieren			Instance.StoragCommit

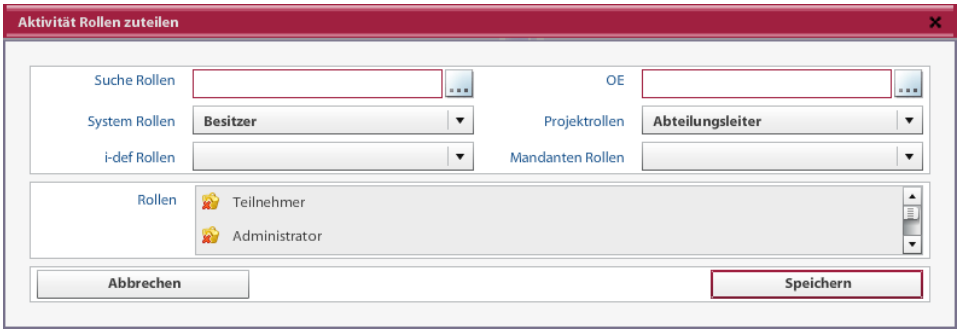
Verknüpfung aktualisie-ren		E445	Link.GetMaxVersion
E-Mail anzeigen		B1	Mail.View
Anzeigen		B480	Object.2Frames
Favorit hinzufügen			Object.AddToFavorites
Eigenschaften anzeigen		B10	Object.Attributes.View
Besitzer wechseln		E115	Object.ChangeOwner
Einchecken		B40	Object.CheckIn
Auschecken		B35	Object.CheckOut
Klassifizieren		B120	Object.Classify
E-Mail senden		E 130	Object.ClientMail
Commit		W 150	Object.Commit()
Projekt Commit		W 333	Project.Commit
Anzeigen		B2	Object.ConstraintObjList
Kopieren		E 085	Object.Copy
Duplikat kopieren		E180	Object.CopyClone
Archivieren		W	Object.Copy2Location
Importieren		B55	Object.CreateAutoImport
Ausschneiden		E 090	Object.Cut
Distribut		W140	Object.Distribute
Lokale Kopie erstellen		E200	Object.Download
Exportieren		E205	Object.Download2Temp
Export wiederholen		E210	Object.DownloadHeader
Ausführen		B 215	Object.Execute
Suchen		B 220	Object.Find
Suchen		B 225	Object.FindInObjList

Verschieben		B0	Object.Move
Referenz nachführen		E12	Object.MoveAliases2ThisVersion
Verknüpfung nachführen		E12	Object.MoveLinks2ThisVersion
Archivieren		W	Object.Move2Location
Version erzeugen		E160	Object.NewVersion
Öffnen		B1	Object.ObjList
Einfügen		B105	Object.Paste
Umbenennen		E110	Object.Rename
Wiederherstellen		B420	Object.Restore
Entsperren			Object,.RestoreStatus
Dossier entsperren		W398	Project.Revert
Änderungen verwerfen		W250	Object.Revert
Zugriff erteilen		E 255	Object.SetAccess
Sichtbarkeit erteilen		E 260	Object.SetShow
Protokoll anzeigen		B	Object.ShowProtocol
Versionen anzeigen		E265	Object.ShowVersions
Signieren		E	Object.Sign
Duplikate synchronisieren		B615	Object.Synchronize
Sammelordner		B615	Object.ToggleCart
Nachführen		E280	Object.Update
Importieren		B 060	Object.Upload
Signatur prüfen		B615	Object.Verify
Anzeigen		B 006	Object.View
Codes produzieren		E	Object.Xml2Ext

Patient zusammenführen			Patient.Merge
Dossier archivieren			Project.Move2Location
SMS senden		B615	Tool.SendSMS
Zusammenführen			Tree.Merge
Admin Passwort ändern		E495	User.AdminPassword
Passwort ändern		E475	User.ChangePassword
Benutzer deaktivieren		E 500	User.Deactivate
Eigenschaften bearbeiten		B460	User.Properties.Edit
Eigenschaften anzeigen		B455	User.Properties.View
Benutzer aktivieren		E 470	User.Reactivate
Umbenennen		B 465	User.Rename
Fälle zusammenführen		B415	Visit.Merge

Rollen zuteilen

Über das Kontextmenü **Aktivität Rollen zuteilen** der betreffenden Aktivität rufen Sie die Maske für die Rollenzuteilung auf. Markieren Sie alle Rollen, für welche die Aktivität verfügbar gemacht werden soll.



Hinweise

- Die Auswahlliste für die Rollen ist von der verwendeten Methode abhängig. Bsp. bei *Object.CheckIn()* steht nur «Aktueller Bearbeiter» zur Verfügung
- Wenn Sie eine Methode zur Erstellung eines neuen Objektes verwenden (z.B. *Object.Create()*), müssen Sie unbedingt im Feld *Neuer Status* den anfänglichen Prozessstatus des neuen Dokuments festlegen.
- Der Besitzer eines Objektes hat immer die vollständigen Rechte auf dem Objekt. Um jedoch an das Objekt zu gelangen, muss er einen Minimalzugriff auf den entsprechenden Ordner haben.

Aktivitätsgruppen festlegen

Um den Aufwand bei der BPM-Diagramm Erstellung zu reduzieren und übersichtlichere Darstellungen zu erhalten, werden sog. Aktivitätsgruppen eingesetzt. Dabei werden Aktivitäten zu einer Aktivitätsgruppe zusammengefasst und im jeweiligen Prozessstatus bestimmten Rollen zur Verfügung gestellt. Gemeinsam ist allen Aktivitäten einer Aktivitätsgruppe, dass kein Statuswechsel ausgelöst wird. Alle Aktivitäten mit Statuswechsel müssen im BPM separat dargestellt und definiert werden.

Im *i-engine* Standard sind folgende Aktivitätsgruppen definiert:

@View pe

- Object.View
- Mail.View
- Object.Download
- Object.ClientMail
- Object.Attribute.View
- Object.AddAlias
- Object.AddLink
- Object.CreateAlias
- Object.RemoveLink
- Object.ShowVersion
- Object.ShowProtocoll

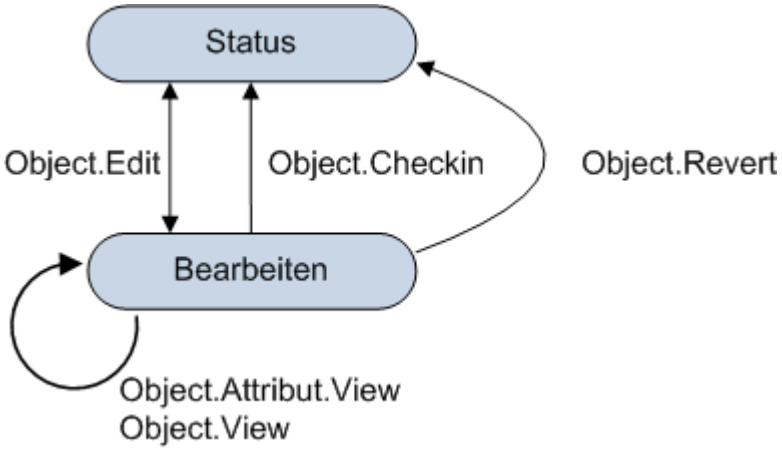
@Edit pe

- Object.Edit
- Object.Attribute.Edit
- Object.Checkout
- Object.Checkin
- Object.NewVersion
- Object.CreateAutoImport
- Project.BulkAttributesEdit
- Object.classify

Hinweis

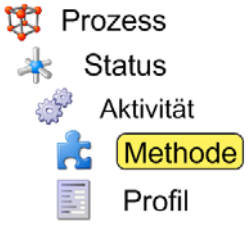
- Änderungen oder neue Aktivitätsgruppen werden im Systemdesign festgehalten

- Eine Aktivitätsgruppe kann auch komplexe Aktivitäten enthalten wie:
  Verwendungsnachweis
  Berechtigte Anzeigen
- Falls Rückwärtskompatibilität bei den Clients erforderlich ist, dürfen max. 12 Aktivitäten in einem Status verfügbar sein. Die Anzahl von 20 Aktivitäten sollte jedoch nicht überschritten werden.
- Eine Aktivitätsgruppe sollte nur Aktivitäten umfassen, die zu keinem Statuswechsel führen. Ausnahme ist die Methode *Object.Edit(),* vorausgesetzt die vorgesehene Behandlung entspricht dem TIE-Standard.



Ggf. sind die Rollenzuteilungen festzuhalten.

Die Methode

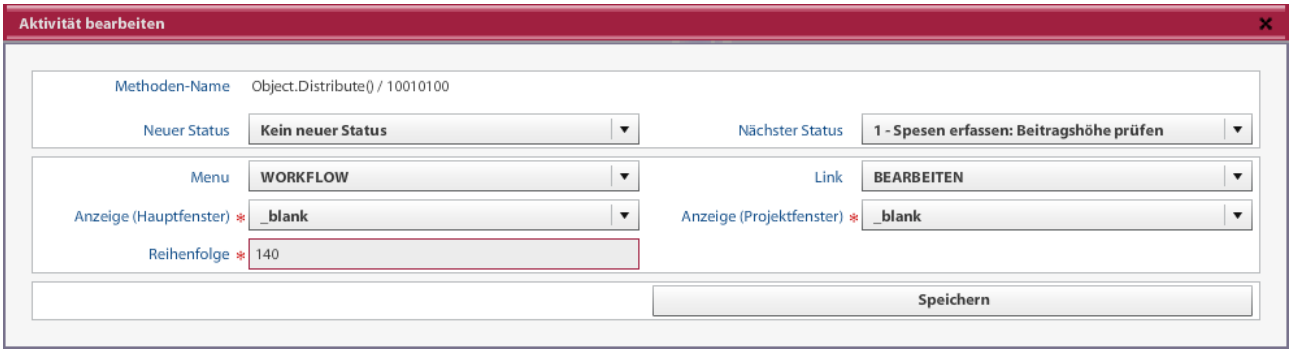


Die eigentliche Funktionalität von *i-engine* wird über so genannte Methoden implementiert. In den Methoden sind die Funktionen ausprogrammiert, welche in den Aktivitäten verwendet werden. Es steht ein umfangreicher Katalog von Methoden zur Verfügung, welcher in einem separaten Abschnitt dokumentiert ist.

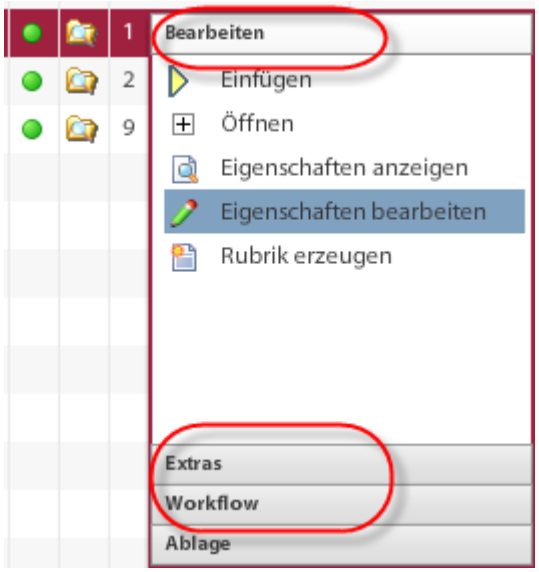
- In jeder Aktivität kann genau eine einzige Methode verwendet werden.
- Wenn Sie mehrere Methoden innerhalb einer einzigen Aktivität benötigen, besteht die Möglichkeit, über die Einstellung in *Nächste Aktivität/Methode* die folgende Aktivität resp. Methode festzulegen. Diese wird nach Beendigung der ursprünglichen Methode automatisch als nächste ausgeführt.

Die benötigte Methode wird auf der ersten Maske der Aktivitäten-Konfiguration bestimmt. Nachdem die Eingabe der Aktivitäts-Parameter mit der Schaltfläche **Weiter** abgeschlossen wurde, wird die Eingabemaske für die Methoden-Parameter angezeigt.

Die Methoden-spezifischen Parameter sind im «Methoden Referenz Handbuch» beschrieben.



Feld	Beschreibung
Methodenname	Ausgewählte Methode. Ob und was bei den beiden Eingabefeldern eingegeben werden muss, hängt von der verwendeten Methode ab. Nähere Angaben dazu finden Sie im Anhang bei den Methoden-Beschreibungen.
Vorlage/Zielordner	Methoden wie *Object.Create()* benötigen eine Objektklasse als Vorlage für das Erstellen eines neuen Objektes.

Feld	Beschreibung
Zwischenstatus	Bei einigen Methoden nimmt das Objekt einen Zwischenstatus ein. Erst wenn die Methode beendet wird, geht das Objekt in den definierten nächsten Status. Bei *Object.CreateEdit()* wird das Dokument direkt zum Editieren geöffnet. Während das Dokument zum Editieren geöffnet ist, nimmt es den hier definierten Zwischenstatus ein. Dieser Status muss natürlich vorher definiert worden sein. Hinweis: *Object.CreateEdit* wird bei neuen Projekten nicht mehr eingesetzt.
Nächster Status	Bestimmen Sie, welchen Status der Prozess nach dem Ausführen der Aktivität einnimmt. Es werden nur die bereits für den Prozess definierten Prozesszustände angezeigt. Natürlich kann auch KEINE Statusänderung erfolgen. Im Statustyp «DISTRIBUTION» / «SUB_DISTRIBUTION» muss der Typ des nächsten Status wiederum «DISTRIBUTION» / «SUB_DISTRIBUTION» sein – ausser es wird die Methode Commit() verwendet.
Neuer Status	Einige Methoden ermöglichen das Erzeugen eines neuen Objektes. Für das neue Objekt, welches oft durch einen anderen Ablaufprozess gesteuert wird, muss der Startprozessstatus angegeben werden. Die Auswahl ist vom Prozessumfeld abhängig.
Menu	Die Aktivität kann im Kontextmenü zur Verfügung gestellt werden.  Es sind 3 Kontextgruppen definiert:

Feld	Beschreibung
	«BEARBEITEN» «EXTRAS» «WORKFLOW» Die Einstellung «NONE» wird verwendet, wenn kein Eintrag im Kontextmenü erscheinen soll. z.B. für Aktivitäten die bereits über die Menüleiste und die Schaltflächen im Projektfenster erreichbar sind.
Link	Wird ein Objekt nicht über den Originalpfad zugegriffen sondern über ein ALIAS-Objekt, können andere Kontextmenü Einträge angezeigt werden. So kann z.B. die Verfügbarkeit einer Aktivität eingeschränkt werden. Aktuell wird eine solche Funktionalität eher durch eine Kombination der Berechtigungsstufen realisiert. Es sind 3 Untermenü definiert: «BEARBEITEN» «EXTRAS» «LINK» Sowie «NONE» s. Abschnitt «*Wozu sind Link/Alias Objekte?* (Seite 187)»
Anzeige (Hauptfenster) Anzeige (Projektfenster)	«_blank»: «content»: «executer»: «NONE»:
Reihenfolge	Anzeigereihenfolge im Kontextmenü. Die Reihenfolge muss weder eindeutig, noch lückenlos vergeben werden. Allerdings sind Doppelvergaben zu vermeiden, da die Reihenfolge innerhalb einer solchen Gruppe zufällig ist.

Protokollierung

Die Protokollierung für diese Aktivitäten wird durch das Feld *Nächster Status* bestimmt. Wenn ein neuer Status eingenommen wird, ist immer ein Wert definiert, und der Vorgang wird protokolliert. Bei Aktivitäten ohne Statusänderung kann aber die Protokollierung ein-/ausgeschaltet werden:

0: Keine Protokollierung
ID des aktuellen Prozessstatus: Vorgang wird protokolliert

Ereignisse - Event Handling

Als Event versteht man ein Ereignis, das sowohl von aussen als auch vom System selbst (z.B. Änderungsbenachrichtigungen) ausgelöst werden kann. Ein Ereignis kann Auslöser (Trigger) für eine Ereignisbehandlung (Event Handling) sein, mit der das System reagiert. Die ereignisgesteuerte Architektur von *i-engine* ermöglicht die Realisierung sehr komplexer und stark automatisierter Geschäftsprozesse oder ganzer Prozessketten.

Das Ereignis umfasst jeweils drei wichtige Angaben

- Auslösendes Objekt (aktuelles Objekt im Geschäftsprozess)
- Art des Ereignisses
- Ereignisbehandlung (Aktivität im gleichen oder einem anderen Prozess auslösen)

i-engine unterstützt die folgenden Ereignisarten

Ablauf Mitteilungszeit (On Notification)

Für den Prozessstatus wird eine Verweilzeit definiert, nach deren Ablauf ein Ereignis ausgelöst wird. Eine synchrone Verarbeitung der Ereignisbehandlung macht keinen Sinn. Beispiel: Hinweismeldung zu einer immer noch pendenten Bearbeitung.

Bei einem Object.Distribute() kann die Mitteilungszeit auch durch die Aktivität vorgegeben werden. Der Auslösende kann damit die Bearbeitungszeit individuell vorgeben.

Ablauf Verfallszeit (On Expiration)

Für den Prozessstatus wird eine Verweilzeit definiert, nach deren Ablauf ein Ereignis ausgelöst wird, welches den Geschäftsprozess fortführen soll. Eine synchrone Verarbeitung der Ereignisbehandlung macht keinen Sinn.
Beispiel: Automatische Freigabe, wenn keine manuelle Reaktion erfolgt ist.

Bei einem Object.Distribute() kann die Verfallzeit auch durch die Aktivität vorgegeben werden. Der Auslösende kann damit die Bearbeitungszeit individuell vorgeben.

Status einnehmen (On Status Entry)

Dieses Ereignis wird ausgelöst, wenn irgendeine Aktivität den betreffenden Status erreicht. Die vom Ereignis ausgelöste Aktion kann synchron oder asynchron abgearbeitet werden. Bei einer synchronen Abarbeitung wird jene Aktivi-

tät, welche zum Statuswechsel geführt hat, so lange verzögert, bis die Ereignisbehandlung abgeschlossen ist.
Beispiel: Nachführen einer Referenz nachdem ein Projekt auf eine andere Ablage verschoben wurde.

Status verlassen (On Status Exit)

Wird ausgelöst, wenn irgendeine Aktivität den Status verlässt. Bei einer synchronen Abarbeitung wird jene Aktivität, welche zum Statuswechsel geführt hat, erst ausgeführt wenn die Ereignisbehandlung abgeschlossen ist.

Aktivität beenden (On Activity End)

Bevor die Aktivität abgeschlossen wird, wird eine Ereignis ausgelöst. Bei einer synchronen Abarbeitung wird die Aktivität komplett ausgeführt aber erst abgeschlossen, wenn auch die Ereignisbehandlung abgearbeitet ist.
Beispiel: E-Mail zur Benachrichtig versenden.

Aktivität auslösen (On Activity Start)

Bevor die Aktivität ausgeführt wird eine Ereignis ausgelöst. Bei einer synchronen Abarbeitung wird die Aktivität erst ausgeführt wenn das Ereignis abgearbeitet ist.

Aktivität auslösen (On Object Setshow Remove Role)

Wird über die Methode *setShow()* die Rollenzuweisung zu einem Objekt gelöscht, wird dieses Ereignis ausgelöst.

Fehler, Abbruch,...

Wird ausgelöst, wenn im Prozess ein Fehler auftritt oder eine Abbruch-Situation eintritt. Keine synchrone Verarbeitung.

Hinweis
Die Definition der Ereignisse erfolgt im Prozessstatus - s. «*Einen neuen Prozessstatus definieren* (Seite 65)» oder in der Definition des IPCs - s. «*IPC - Interprozess Kommunikation* (Seite 243)».

Reihenfolge der Ereignisse

Die Ereignisse treten untereinander in folgender Reihenfolge auf:

1. Aktivität auslösen
2. Status verlassen
3. Status einnehmen
4. Aktivität beenden

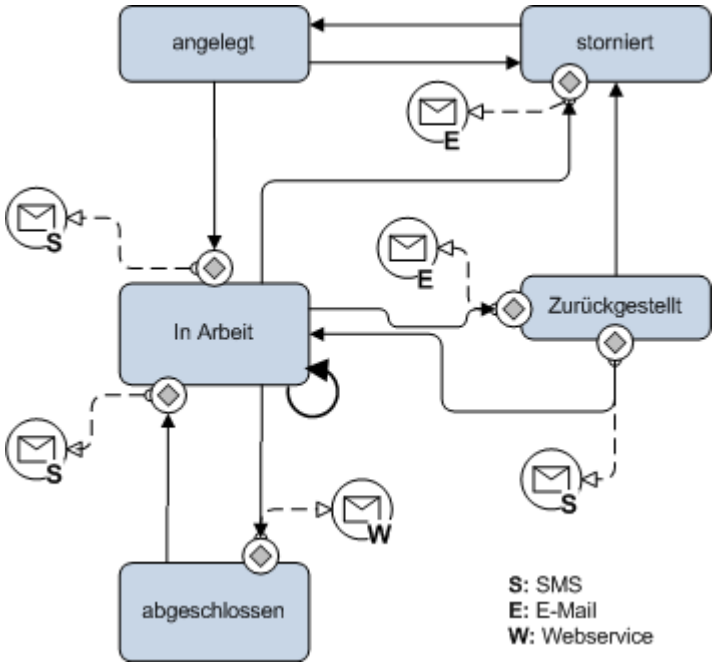
Unified Messaging System - UMS

Das Unified Messaging System der *i-engine* stellt einen einheitlichen Mechanismus für den Austausch von Daten zur Verfügung. Diverse Services implementieren die entsprechenden Technologien für den Versand von E-Mails und SMS-Meldungen, den Datenaustausch via SOAP oder mittels Datei. Im Geschäftsprozess werden die Meldungen über Ereignisse s. «*Event Handling* (Seite 102)» ausgelöst. Der Benutzer kann für die gewünschten Meldungen ein Abonnement lösen und erhält damit nur die erforderlichen Meldungen übermittelt.

Das Unified Messaging System bearbeitet die Meldungen unabhängig vom Meldungstyp:

- E-Mail
- SMS Meldung
- Webservices, SOAP
- Dateitransfer
- docbox und andere

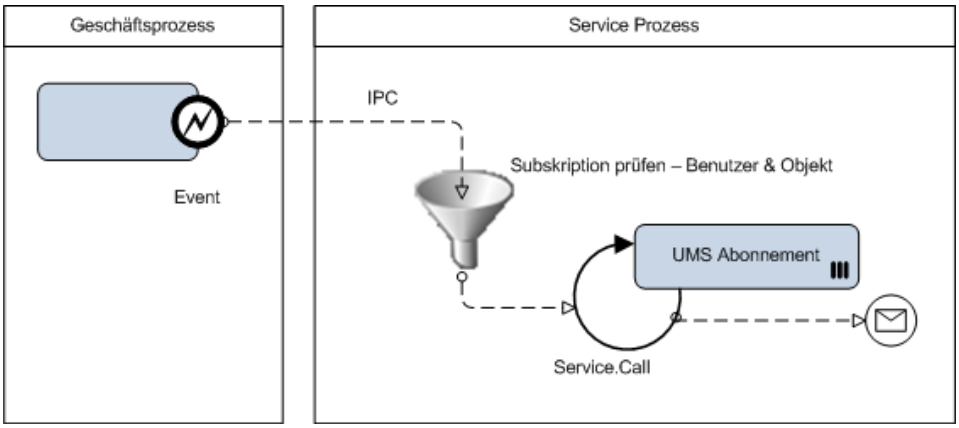
Die im Workflow benötigten Benachrichtigungen und Meldungstypen werden im Systemdesign spezifiziert.



Prinzipieller Ablauf

Das Unified Messaging System besteht aus zwei Grundelementen:

- Das Event Handling löst via IPC den Service-Call zum Senden der vorgesehenen Meldung aus. In einem Abonnement sind die dazu notwendigen Einstellungen - was wird wie versandt - festgehalten.
- Die Abonementsverwaltung prüft, ob der Teilnehmer für das Objekt eine Subskription des betreffenden Abonnementes hat und generiert ggf. die Meldung.



Vorbereitende Massnahmen

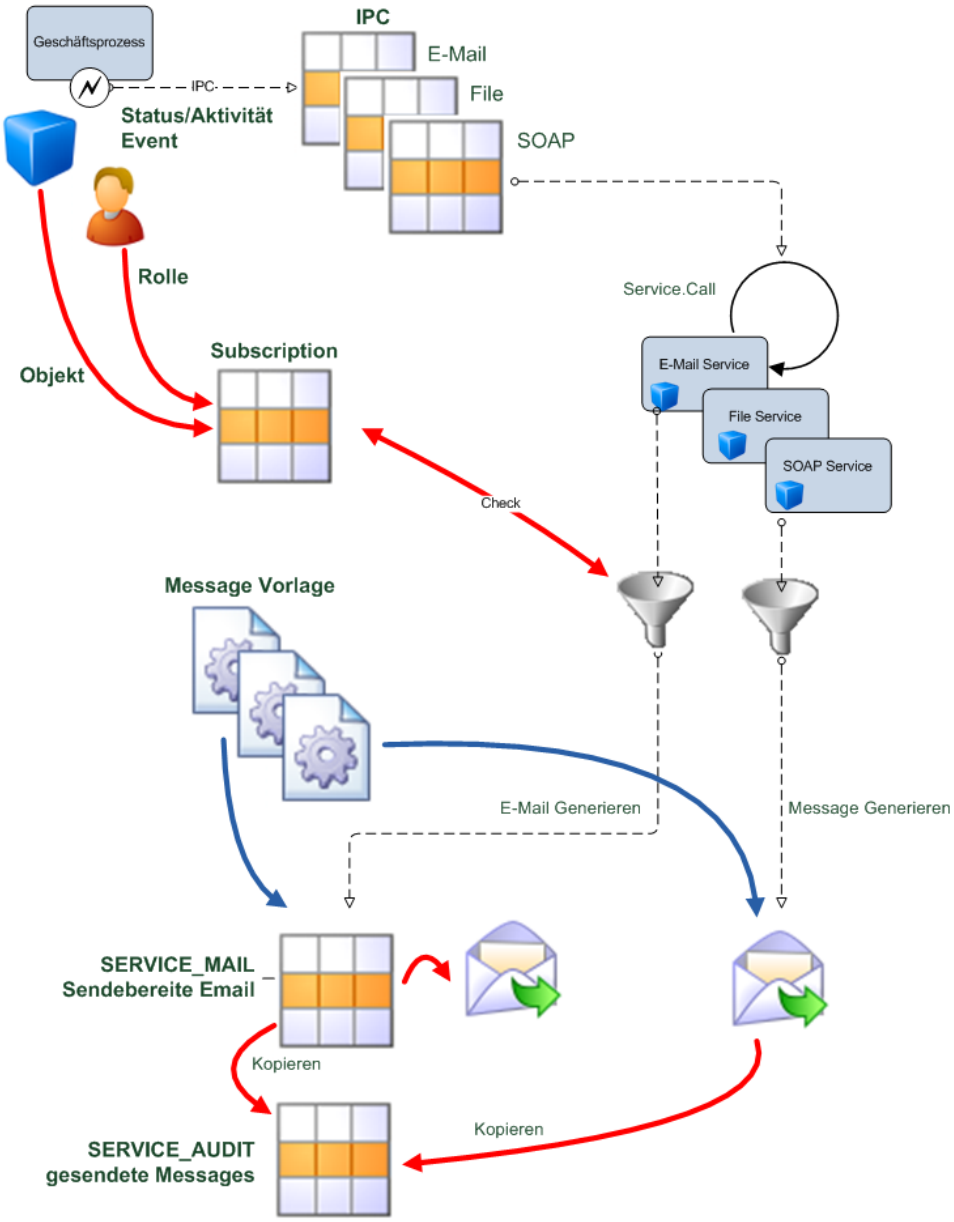
- Definition des Abonnements: Festlegen des Meldungtyps und der Meldungsvorlage
- Subskription für ein Abonnement: Der Benutzer abonniert ein Abonnement für bestimmte Objekte
- Definition der Ereignisse: Für Meldungen bei Ablauf- und Mitteilungsereignissen werden die Wartezeiten im Prozessstatus definiert. Andere Ereignisse werden direkt in den IPC-Einträgen konfiguriert. Beim Auftreten des Ereignisses löst der betreffende IPC die Methode *Service.Call()* auf, um eine UMS-Meldung auszulösen.

Für die Realisierung reiner E-Mail Funktionalität stehen in *i-engine* zusätzlich folgende Mechanismen zur Verfügung:

- E-Mail im Workflow
- E-Mail in der Systemüberwachung

Diese Methoden sind aber bei neuen Implementationen nicht mehr einzusetzen.

Implementierung UMS



Eintrag in IPC Tabelle

DB Action

Für jeden durch das Ereignis ausgelösten Meldungstyp muss ein Eintrag in der IPC Tabelle vorgenommen werden. Weitere Angaben über die IPC Tabelle sind dem Abschnitt «*IPC - Interprozess Kommunikation* (Seite 243)» zu entnehmen.

Feld	Eintrag
EVENT	Ereignis, bei welchem eine Meldung generiert werden soll. «ON_ACTIVITY_START»: «ON_ACTIVITY_END»: «ON_STATUS_ENTRY»: «ON_STATUS_EXIT»: «ON_EXPIRATION»: «ON_NOTIFICATION»: «ON_OBJECT_SETSHOW_REMOVE_ROLE»:
STATUS_ID resp. ACTIVITY_ID	Für die Trigger-Bedingung «ON_ACTIVITY_END» wird die ID der Aktivität angegeben, sonst die ID des Prozessstatus.
CONDITION_ STMT_ID	SQL-Statement welches prüft, ob der Trigger aktiv oder inaktiv ist. «YES» oder «»: Trigger verwenden «NO»: Trigger nicht verwenden
TARGET	Abhängig vom Meldungstyp wird das betreffende Service-Objekt festgelegt. Standardmässig vorhanden sind folgende Service-Objekte: «Sofort E-Mail» - 10080821 «Sammel E-Mail pro Stunde» - 10080822 «Sammel E-Mail pro Tag» - 10080823 «docbox» - 10080581 «Sofort SMS» - 1008829
TARGET_ ACTIVITY_ID	Aktivitäts-ID des gewünschten Service.Call()
TARGET_ METHODE_ID TARGET_ REFERENCE	Methode *ServiceCall()* - 10010228 - zusammen mit TARGET_REFERENCE), um Eindeutigkeit zu erreichen. Wird verwendet, wenn TARGET_ACTIVITY_ID nicht aufgesetzt ist. Die Spezifikation mit der Methode ist jener mit einer Aktivität vorzuziehen, da keine applikationsspezifischen IDs verwendet werden.
TARGET_ PARAMS	Parameter für den Aufruf den *ServiceCall()* Methode
DUE_DATE	Festlegen, wann die Trigger-Aktion durchgeführt werden soll.

Feld	Eintrag
	«Null»: sofort, synchron «0»: sofort, asynchron «n»: Anzahl Tage ab heute; Bsp. 1.5 => 36h
REPEAT	Anzahl Wiederholungen Der IPC kann in bestimmten Zeitintervallen wiederholt werden, solange seine Auslösebedingungen erfüllt sind. Das Zeitintervall ist durch den Wert in DUE_DATE vorgegeben.
SEQ	Abarbeitungsreihenfolge, wenn mehrere Trigger für die gleiche Situation definiert sind.

Hinweis
Mit einem Param SQL-Statement werden die aktuellen Parameter an die *Service.Call()* Methode übergeben; bspw. die Vorlage oder der Anhangtyp.

Eintrag in Subscription Tabelle

DB Action

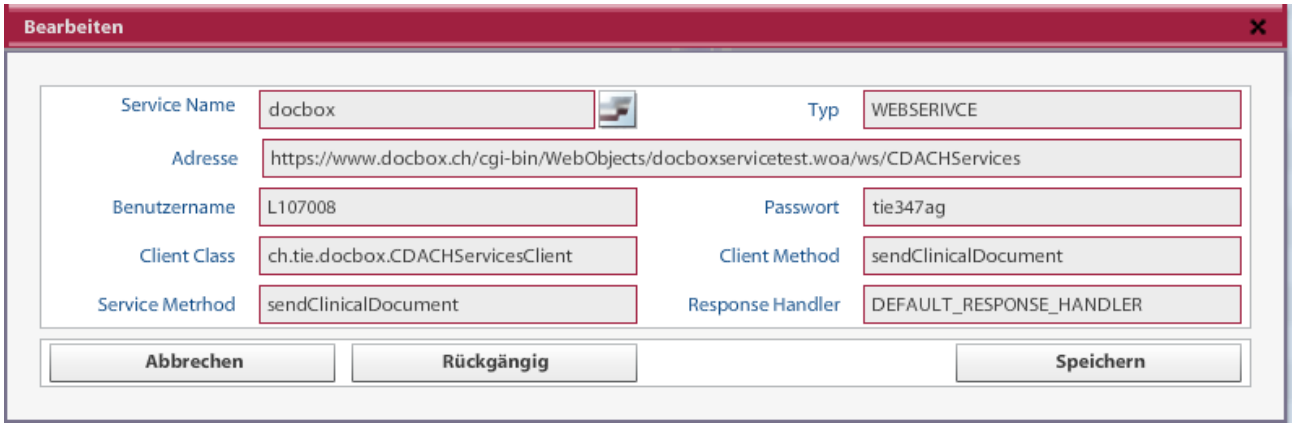
Das Abonnement definiert die Empfänger einer bestimmten Meldung und welcher Mechanismus zum Einsatz kommt.

Feld	Eintrag
OBJ_ID	Startobjekt, ab welchem das Abonnement aktiv ist.
ROLE_ID	Falls dies ein individueller Eintrag ist, wird die persönliche Rolle eingesetzt.
SERVICE_ID	Zuständiger UMS-Service
RECI-PIENT_NR	Im User-Objekt können für den Service (aktuell nur E-Mail) mehrere Adressen festgelegt werden. «1..3»: Auswahl des gewünschten Eintrages (Email1..Email3)
TREE	«YES»: Alle untergeordneten Objekte sind abonniert und dürfen dem Benutzer E-Mails senden.

Feld	Eintrag
	«NO»: Nur das aktuelle Objekt
RESTRICTION	Beenden der Aboeinstellung für eine Rolle ab diesem Objekt. Gibt es einen Abo-Eintrag mit TREE auf «YES» gesetzt, werden alle untergeordneten Elemente auch berücksichtigt. Durch einen zusätzlichen RESTRICTION Eintrag kann das Abonnement für einen Baumzweig wieder gelöscht werden. «YES»: Eintrag definiert ein Stopp-Kriterium in der Baumstruktur. d.h. Abonnement ab da gelöscht. «NO»: Eintrag definiert ein Abonnement.

Eintrag in Service Tabelle

Für jeden durch das Ereignis ausgelösten Meldungstyp muss ein Eintrag in der IPC Tabelle vorgenommen werden. Weitere Angaben über die IPC Tabelle sind dem Abschnitt «*IPC - Interprozess Kommunikation* (Seite 243)» zu entnehmen.



Feld	Eintrag
Service Name	Bezeichnung des Services - mehrsprachig
Adresse	URL auf den Provider/Dienst
Benutzername	Oft wird eine Authentifizierung beim Provider/Dienst eingesetzt
Passwort	
Type	Dienst-Art «SOAP» -> Webservice «JMS» -> Java Messages Service «EMAIL» «FILE» -> Text/XML datei
Client Class	Die Implementierung in *i-engine* erfolgt über eine Java-Klasse. Diese muss entsprechend der Dienst-Art aufgesetzt werden. s. Referenzhandbuch «Service.Edit()»
Default Client Methode	Definiert die Methode, die auf der Client Class (Proxy) aufgerufen werden soll.
Default Service Methode	Definiert die Methode, die auf dem Remote Service aufgerufen werden soll.
Response handler	PL/SQL Funktion, welche die Antwort der Service_Call Methode bearbeitet.
Adress Typ	«puser.mobile1» «puser.user_name» «puser.email» In der Subkription wird zusätzlich festgelegt, welcher E-Mail Eintrag verwendet werden soll.

Auslösen der Meldung

Der IPC führt die Methode *Service.Call()* aus, um die Meldung auszulösen. Über ein Param SQL-Statement werden folgende Parameter für die Methode aufgesetzt:

Parameter	Beschrieb
template_obj_id	Vorlagenobjekt Je nach Servicetyp müssen entsprechende Vorlagen vorbereitet werden: «EMAIL»: E-Mail mit Platzhaltern «FILE»: Dateigerüst mit Platzhaltern «SOAP»: XML-Definition des Datenaustausches
abo	Auswertung der Subscription vornehmen: «NO»: Empfänger aus den Parameterdaten bestimmen «YES»: Empfänger sind durch Subscription-Einträge definiert
recipient_member_type	Dynamische Systemrollen Diese werden nur berücksichtigt, wenn kein Abonnement verwendet wird. «MEMBER_TO» «MEMBER_CC»:
recipients	Empfängerliste Diese wird nur berücksichtigt, wenn kein Abonnement verwendet wird. Mehrere Einträge werden durch Kommas separiert. die Angabe ist vom Servicetyp abhängig. Bsp. EMAIL: «test@google.com»
recipients_roles	Empfängerrollen Diese werden nur berücksichtigt, wenn kein Abonnement verwendet wird.
data_attachment	Aktuelles Objekt als Anhang mitschicken «NONE»: Nein «VIEW»: Datei der View-erweiterung «APPLICATION»: Datei der Application-Erweiterung

Empfänger festlegen:

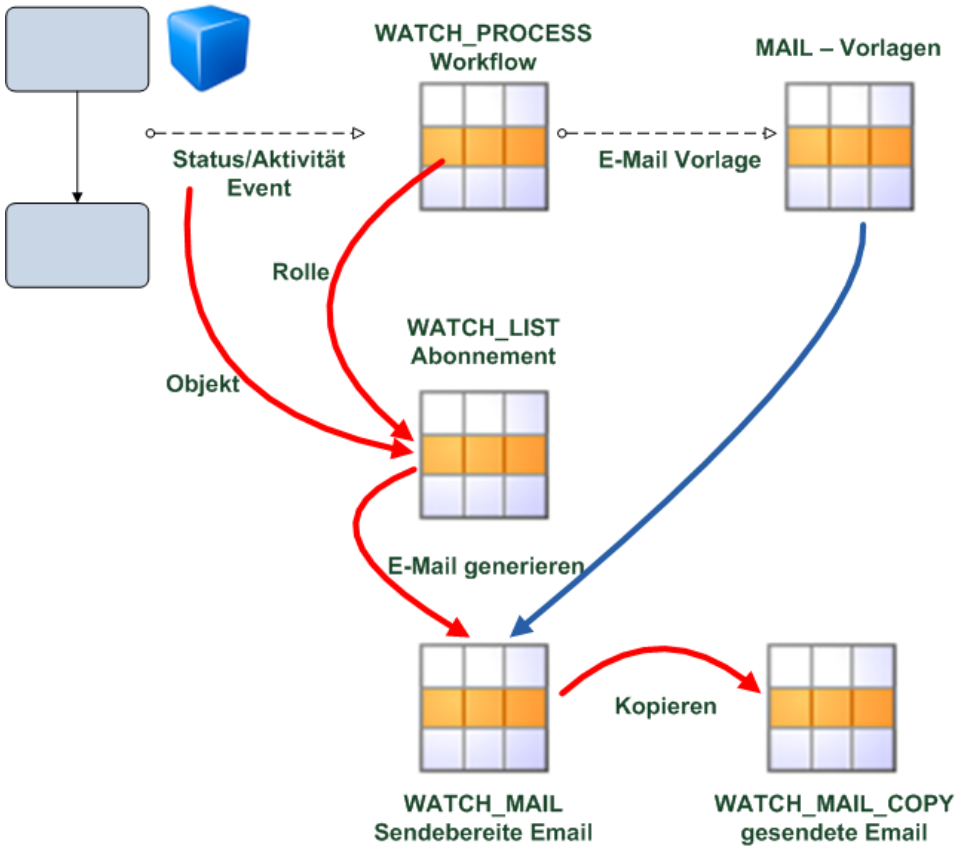
- durch die Festlegungen in der Subcription Tabelle.
- direkte Übergabe der Empfängeradresse *recipients* im Param SQL-Statement.
- Übergabe der Rollen *recipients_roles, recipient_member_type* im Param SQL-Statement. Wird oft eingesetzt, wenn vorgängig ein SetShow() die Berechtigten aufgesetzt hat.

Die aktuelle Implementation hat folgende Einschränkungen:

- Ein bestehender Eintrag kann überschrieben, aber nicht gelöscht werden.
- Mehrfacheinträge für die gleiche Kombination aus Rolle und Objekt sind nicht möglich.
- Für die Anzeige/Abfrage bereits definierter Subskriptionen gibt es keine Methode.

E-Mail Versand (veraltet)

E-Mails im Workflow



Mechanismus

- Die vorgesehene E-Mail ist als Vorlage in der Tabelle MAIL abgelegt
- Ein Eintrag in der Tabelle WATCH_PROCESS definiert die Prozessbedingung, wann die betreffende E-Mail ausgelöst werden soll. Über die angegebene Rolle sind die potentiellen Adressaten definiert.

- In der Tabelle WATCH_LIST ist bestimmt, wer bei welchem Objekt eine E-Mail erhalten soll.
- Entsprechend wird unter Verwendung der Vorlage eine E-Mail für jeden Adressaten in WATCH_MAIL angelegt. Diese werden beim nächsten Durchlauf des «Watchers» verschickt.
- Verschickte E-Mails werden gelöscht und dafür zur Kontrolle in der Tabelle WATCH_MAIL_COPY eingetragen.

E-Mail Vorlage

DB Action

Das E-Mail-Objekt enthält die Definition des Betreff- und Meldungstextes. Aktuell wird der eigentliche Inhalt noch direkt in der Tabelle MAIL festgelegt.

E-Mail-Objekt erzeugen

Für eine neue E-Mail muss ein E-Mail-Objekt – Objektklasse SMTP-Email / Objektklassentyp MAIL – erstellt werden. Dies geschieht am einfachsten, durch das Kopieren einer bereits vorhandenen E-Mail.

Meldungstexte erfassen

In der Tabelle MAIL sind die folgende Eingaben für jede E-Mail anzupassen:

Feld	Bedeutung
SUBJECT	Betrefftext - mehrsprachig
CONTENT	Meldungsinhalt - mehrsprachig
CONTENT_TYPE	«Text/html; charset="windows-1252"» Zwingend vorgeschrieben, damit Links auch im WinClient und/oder FlexClient funktionieren können «Text/plain; charset="windows-1252"» nur für reine HTML-Anwendungen Bei einem leeren Feld wird der entsprechende Wert in einer E-Mail automatisch aufgesetzt, abhängig von der Grundeinstellung des Systems (Eintrag in der Tabelle SYS_PROPERTIES key = WATCHERCONTENT_TYPE, value = TEXT oder HTML. Wenn der Eintrag fehlt, wird TEXT angenommen.

In den beiden Texten können auch Variablen eingesetzt werden. Hier die Aufstellung der verfügbaren Variablen:

?{object.xxxx}	Über «object» können alle Felder der Objektklasse DOCUMENT referenziert werden.
?{object. next_expiration_date}	Wird bei einem *Distribute()* verwendet:

	Während «object.expiration_date» ggf. das Datum aus der letzten Aktivität enthält, entspricht «object.next-expiration_date» dem Datum der aktuellen *Distribute()* Aktivität.
?{object. next_notification_date}	Wird bei einem *Distribute()* verwendet: Wie beim obigen Feld.
?{object.obj_id}	Objekt, auf welchem etwas ausgelöst wurde.
?{parent.xxxx}	Mit «parent» stehen die gleichen Felder wie bei «object» zur Verfügung, allerdings für das übergeordnete Objekt. «parent» kann auch mehrfach verwendet werden. ?{parent.parent.t1} resp. ?{parent.parent.parent.t1}
?{activity.activity_id} ?{activity.activity_name}	Ausgelöste Aktivität
?{user.p_user_id} ?{user.fullname} ?{user.user_name}	Benutzerdaten
?{protocol_server_port} ?{url} ?{target_system_descr	Angaben zum Link, abhängig vom verwendeten Client
?{root.xxx}	Attribute des nächsten Root-Objektes. (falls das Objekt selbst ein Root ist, dann sind die Attribute identisch mit ?{object.xxx}
?{patient.xxxx} ?{visit.xxxx}	Healthcare Tabellen

Workflow Verknüpfung

Als nächstes ist im Workflow die Situation festzulegen, wann eine E-Mail verschickt werden soll. Jede Aktivität kann bspw. zum Abschluss einen Versand auslösen. Oder wenn der Prozess einige Zeit in einem Prozessstatus verharrt, kann ebenfalls eine E-Mail verschickt werden.

In der Tabelle WATCH_PROCESS wird die Zuordnung zwischen Prozessstatus/Aktivität, der zusendenden E-Mail und den Empfängern bestimmt.

Activity_ID	Das aktuelle Objekt befindet sich in einem Prozessstatus in welchem diese Aktivität zur Verfügung steht. Der eingetragen Wert ist vom «Event> abhängig. Falls am Ende einer Aktivität «ON_ACTIVITY_END» eine E-Mail erzeugt werden soll, ist die zugehörige Aktivität festzuhalten.

	Für Meldungen nach Ablauf einer bestimmten Zeit (Status Ereignisse «ON_NOTIFCATION» oder «ON_EXPIRATION) muss dieses Feld auf «0» gesetzt werden.
Status_ID	Die Zuordnung zwischen Aktivität und Prozessstatus ist eindeutig. Auch wenn im vorherigen Feld eine Aktivität eingetragen wurde, muss der korrekte Prozessstatus nachgetragen werden. Für «ON_NOTIFCATION» oder «ON_EXPIRATION» muss der zu überwachende Prozessstatus angegeben werden.
Rollen_ID resp. Role_Stmt_ID	Rolle, deren Mitglieder die E-Mail abonnieren können. Gebräuchlich sind: Teilnehmer – 10030009 Besitzer – 10030010 Die Rollen IDs können dynamisch zum Zeitpunkt der Ausführung bestimmt werden. Das entsprechende ROLE SQL-Statement liefert die Liste der Rollen-IDs und des Membertypes (z.B. MEMBER_TO).
Member_Type	Wenn beim Rollen-ID «Teilnehmer» angegeben wird, muss auch der Member_Type definiert werden Beim *Distribute* werden unterschiedliche Adressatengruppen festgelegt. Über den Member_Type wird festgelegt, für welche dieser Adressatengruppe der Eintrag gilt. «MEMBER_TO»: Teilnehmer «MEMBER_CC»: orientierte Teilnehmer «MEMBER_FU»: verfolgende Teilnehmer «ADDED_MEMBER_TO»: Teilnehmer - neue «ADDED_MEMBER_CC»: orientierte Teilnehmer - neue «REMOVED_MEMBER_TO»: Teilnehmer - gelöschte «REMOVED_MEMBER_CC»: orientierte Teilnehmer - gelöschte Hinweis: ■ Der Eintrag «MEMBER_FU» wird für jene Teilnehmer verwendet, die beim *Distribute()* eingetragen haben, dass sie das Objekt weiter verfolgen möchten. ■ Werden mehrere *Distribute()* in Sequenz verwendet, bleiben die Einträge der «MEMBER_FU» erhalten – und das Mitglied bleibt orientiert. ■ ADDED_ Teilnehmer sind bei späteren Schritten MEMEBR_ Teilnehmer, da sie ja nur beim aktuellen Schritt hinzugeführt wurden. ■ REMOVE_ Teilnehmer sind entsprechend bei späteren

	Schritten nicht mehr dabei.
Mail_ID	Zugehörige E-Mail Vorlage
Event	«ON_ACTIVITY_START»: bei Aktivitätsstart «ON_ACTIVITY_END»: bei Aktivitätsende «ON_STATUS_ENTRY»: beim Eintritt in einen Status «ON_STATUS_EXIT»: beim Verlassen eines Status «ON_NOTIFCATION»: bei Überschreiten der Benachrichtigungsdauer «ON_EXPIRATION»: bei Überschreiten der Ablaufdauer
CONDITION_ STMT_ID	SQL-Statement welches prüft, ob die E-Mail aktiv oder inaktiv ist. «YES» oder «»: Trigger verwenden «NO»: Trigger nicht verwenden Anwendung: E-Mail Versand ist von der Auswahl einer bestimmten Option abhängig.
Attachement	Das aktuelle Objekt wird als Anhang der E-Mail angefügt. Über diese Einstellung wird der zu verwendende Extensiontyp festgelegt: «NONE»: Kein Anhang «VIEW»: z.B. pdf Format «APPLICATION»: z.B.doc-Format «PRINT»: z.B. pdf-Format

E-Mails Abonnieren

Die Rollenmitglieder erhalten nur jene E-Mails zugeschickt, die sie explizite abonniert haben. In der Tabelle WATCH_LIST wird für jeden Benutzer der Startknoten festgelegt, ab welchem Workflow-Objekt ggf. E-Mails an ihn geschickt werden sollen.

USER_NAME	Einträge sind für jeden Benutzer individuell
OBJ_ID	Startobjekt, ab welchem das Abonnement aktiv ist. «0»: Kein Objekt im Workflow zugeordnet, die E-Mail

	wird via Systemüberwachung verschickt; s. Abschnitt «*Systemüberwachung* (Seite 121)»
STRUCTURE	«YES»: Alle untergeordneten Objekte sind abonniert und dürfen dem Benutzer E-Mails senden. «NO»: Nur das aktuelle Objekt
STMT-ID	Bei E-Mails im Workflow: «1»: fix aufgesetzt Hinweis: SQL-Statement mit ID=«1» vom Typ «Watch_Process» ist hinterlegt. Andere Werte: s. Abschnitt «*Systemüberwachung* (Seite 121)»
EMAIL1 .. EMAIL3 MOBILE1	Jeder Benutzer hat 3 E-Mail-Adressen, sowie eine SMS-Adresse.
TEXT1 .. TEXT4	Bemerkung: wird nicht mehr verwendet
INTERVALL	«IMMEDIATE»: Die Meldung wird sofort an den Benutzer weitergeleitet. «1h», «24h» Die Meldung werden über das Zeitintervall gesammelt, und erst am Ende dem Benutzer weitergeleitet.

Hinweis

- Über die Vergabe des Startobjektes kann sehr einfach festgelegt werden, dass bestimmte E-Mails nicht an externe Stellen gehen sollen. Die Objekte können dazu einer internen und die freigegebenen zusätzlich einer externen Struktur zugewiesen werden. Wird den externen Benutzern das Wurzelobjekt der externen Ablagestruktur zugewiesen, erhalten Sie nur E-Mails zu freigegebenen Dokumenten.

Versenden nach Ablauf

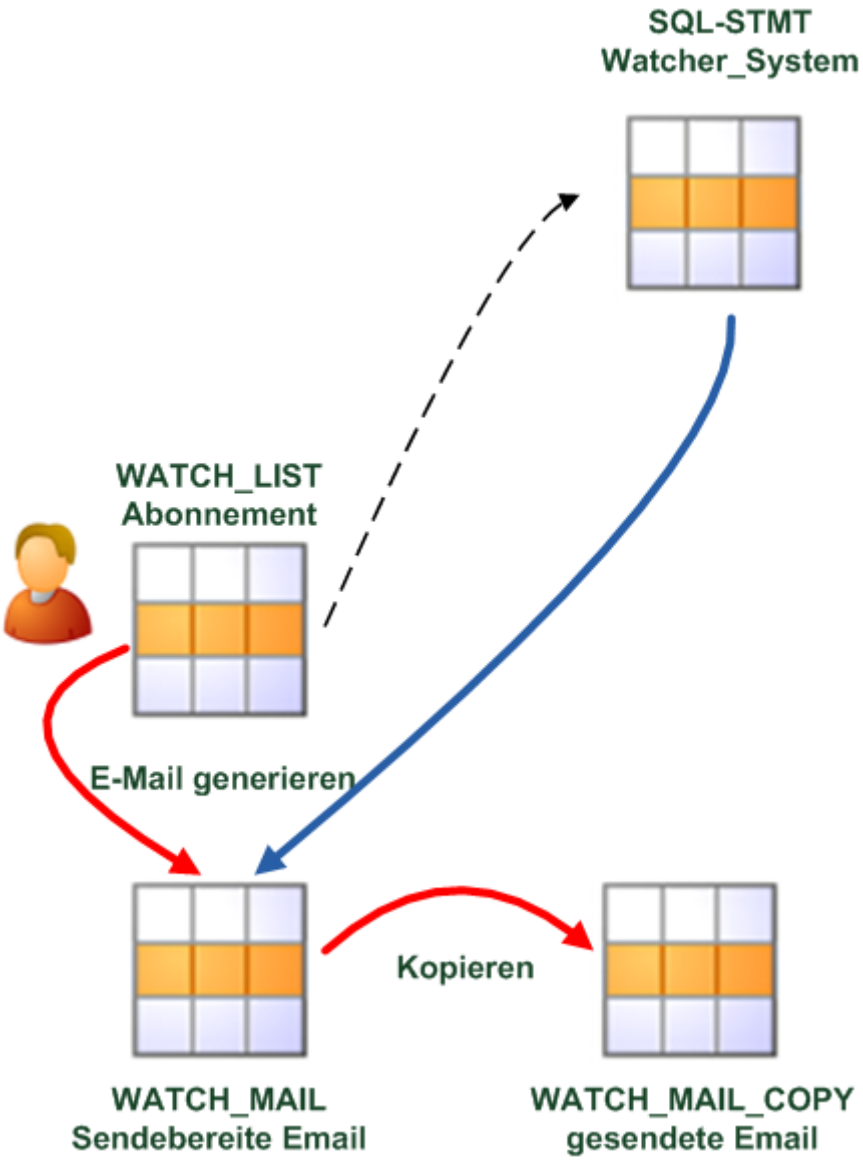
Wenn ein Status mit automatischen Benachrichtigungs- oder Ablaufmeldungen erreicht wird, wird ein Systemprozess auf den betreffenden Zielzeitpunkt eingeplant. Sollte der Status mehrfach erreicht werden, wird jedes Mal ein weiterer Systemprozess eingeplant. Zum Zeitpunkt des Versands wird geprüft, ob die seinerzeitigen Bedingungen noch erfüllt sind, und ggf. die Benachrichtigung verschickt.

Eingeplante Benachrichtigungen werden nicht gelöscht, wenn z.B. der Status verlassen wird, sondern sie werden nicht mehr ausgeführt, weil die erforderlichen Bedingungen u.U. nicht mehr gegeben sind.

Hinweise

- Bei einem neuen Benutzer werden die Abonnement-Einstellungen aus der Benutzervorlage übernommen.
- Der Mailprozess muss entsprechend den gewählten Zeitintervallen aufgesetzt werden. Andernfalls bleiben die E-Mails liegen, bis der Prozess wieder aktiv wird.
- Für den E-Mail Text ist die Spracheinstellung des Empfängers ausschlaggebend.

E-Mails in Überwachung



Mechanismus

- Der «Watcher» überprüft regelmässig alle Einträge in WATCH_LIST, welche nicht für den Workflow eingetragen sind (Kein Objekt zugeordnet)
- Ein SQL-Statement dient der Überprüfung der Auslösesituation und liefert ggf. die für die Generierung des E-Mails benötigten Angaben zurück.
- Falls die Auslösesituation erfüllt ist, wird für den Benutzer eine neue E-Mail generiert und in WATCH_MAIL angelegt. Diese werden anschliessend verschickt.
- Verschickte E-Mails werden gelöscht und dafür zur Kontrolle in der Tabelle WATCH_MAIL_COPY eingetragen.

Parametrisieren E-Mails in Überwachung

DB Action

Ein Reihe von benötigten E-Mails sind nicht an den Workflow gekoppelt. Sie werden über den Mechanismus der «Systemüberwachung» implementiert.

System-Watcher SQL-Statement

Das System-Watcher SQL-Statement prüft die Voraussetzungen für die Benachrichtigung und liefert ggf. alle Angaben für den Aufbau einer E-Mail-Meldung. Das SQL-Statement muss die folgenden Rückgabewerte liefern:

ObjID	
Subject	Die Betreffangabe für die E-Mail muss in 4 Sprachen zurückgeliefert werden
Text	Der Meldungstext muss in 4 Sprachen zurückgeliefert werden.
link	In der E-Mail kann ein direkter Link auf ein Objekt eingebaut werden. Das SQL-Statement darf nur Links liefern, auf die der E-Mail-Benutzer Zugriffsrechte hat.
content_type	

Der Aufbau des SQL-Statements ist kompliziert. Als Basis sollte die PL/SQL-Funktion C_WATCHER_SYSTEM verwendet werden. In dieser sind die folgenden Ereignisse überwacht:

- Überwachung IMAP: Passwort, Einstellungen, Import, Admin
- Überwachung WebDAV: Import, Admin

Für die Überwachung durchsucht das SQL-Statement die Log-Datei auf entsprechende Fehlereinträge.

Meldungen abonnieren

Um die generierten Nachrichten zu erhalten, müssen diese abonniert werden. In der Tabelle WATCH_LIST sind dazu folgende Einträge speziell aufzusetzen:

OBJ_ID	leer
STRUCTURE	«NO»
STMD-ID	ID des SQL-Statement vom Typ «Watcher_System».

Hinweis

- Für die Erzeugung der E-Mail ist der Dienst «WatcherSystem» verantwortlich. Er generiert die E-Mails direkt. Dh. es ist keine Definition eines E-Mail-Objektes notwendig.

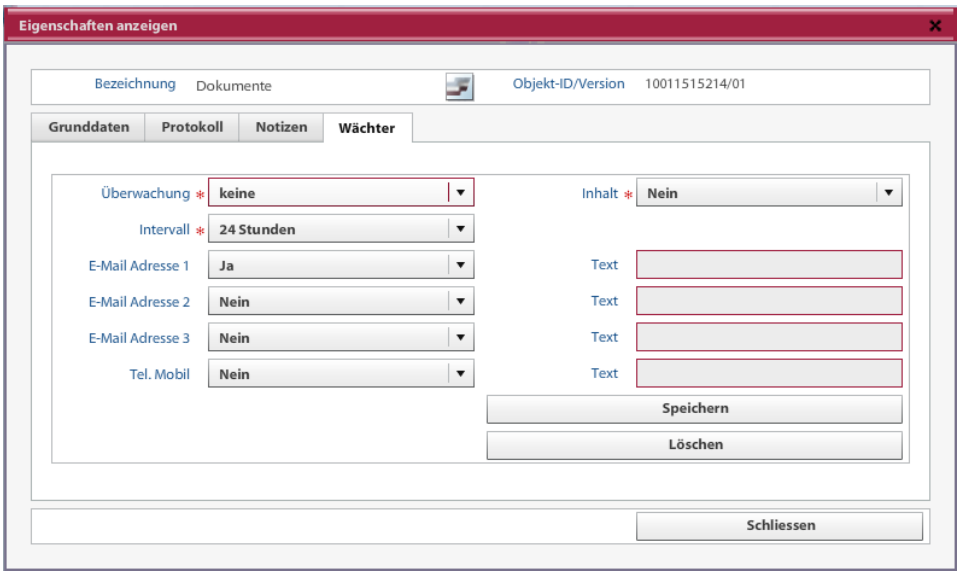
Datenwächter

Über den Datenwächter kann dem Benutzer direkt ein E-Mail zur Orientierung übermittelt werden, wenn sich etwas am Inhalt eines Ordners oder Dokumentes ändert. Bestehende Datenwächter können von jedem Benutzer auf jedes Objekt angewandt werden; vorausgesetzt sie haben Schreibzugriff auf die Eigenschaften des Objektes.

Jeder Datenwächter überprüft regelmässig, ob es Objekte hat, welche die Überwachungskriterien erfüllen. Alle Benutzer, die eine Überwachung für ein Objekt in der resultierenden Liste definiert haben, werden anschliessend informiert.

Jeder Benutzer kann die gesetzten Überwachungen in einem speziellen Arbeitsordner auflisten und kontrollieren.

Welche Ablage-, Eingangs-, Arbeitsordner oder Dokumente durch den Datenwächter beobachtet werden soll und welche Veränderungen - z.B. neues Protokoll abgelegt - eine Meldung auslösen sollen, kann frei definiert werden. So könnte z.B. der Betreffende jeweils ein E-Mail erhalten, wenn ein neues Dokument visiert werden muss.



Hinweis

- Der Datenwächter steht dem Benutzer zur Verfügung, wenn im Attributprofil der entsprechende Registereintrag vorgenommen wird. s. Abschnitt «*Eingabemasken / Maskenaufbau* (Seite 145)».

- Das SQL-Statement mit ID 10001166 vom Type WATCHER_OBJSPECIFIC implementiert die Standard Änderungsüberwachung. Die Anwendung sollte normalerweise keine weiteren Datenwächer einsetzen.

Eingabemasken und Listenanzeige

Die Definition von Eingabe- oder Bestätigungsmasken und die Anzeige von Inhaltslisten im Datenfenster werden durch Attributprofile gesteuert. Der Standard von *i-engine* enthält für jede Methode ein eigenes Standard Attributprofile. Diese vorhandene Auswahl kann mit weiteren Profilen für spezifische Masken und Listen erweitert werden.

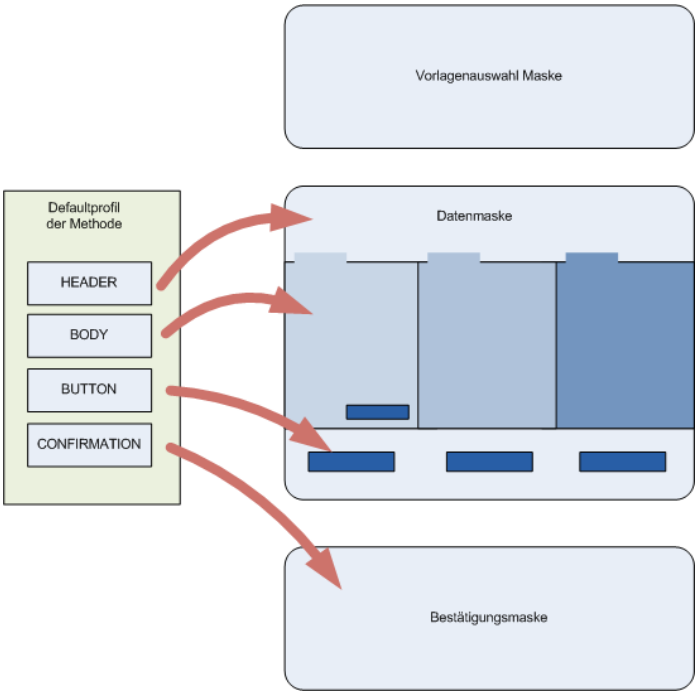
Die Attributlisten werden jeweils einer Aktivität zugeordnet, wo sie für die Formatierung der notwendigen Ein-/Ausgaben verwendet wird. Je nach Methode enthält ein Profil Angaben für bis zu 3 Masken: Vorlagenauswahl, Datenmaske und Bestätigungsmaske.

Hinweis

Im Prinzip verwendet die Aktivität für alle Objektklassen das gleiche Attributprofil. Allerdings können für einzelne Objektklassen Abweichungen in der Tabelle WF_ACTIVITY_PROFILE festgelegt werden. s. auch Abschnitt «*AP-Reference (Seite 77)*».

Maskenaufbau

Im Attributprofil werden die einzelnen Teilbereiche für den Aufbau der Masken abgelegt. Für fehlende Teile wird auf die Vorgaben aus dem Methoden Default-profil zurückgegriffen.



Hinweis

- Aktuell werden Attributprofile direkt in der Datenbank definiert.
- Der Definitionsteil CONFIRMATION wird nur für den HTML Client verwen-det.

Nomenklatur

Für die Behandlung der Daten einer Objektklasse werden mehrere Attributprofile benötigt, die sich je nach Anwendungszweck mehr oder weniger unterscheiden. Die Bezeichnung des Attributprofiles wird aus dem aktuellen Modulnamen, der Objektklasse und dem Methodennamen zusammengesetzt. Es werden die Kurzbezeichnungen (Mnemonics) verwendet.

«Modul»_«Objektklasse»_«Methode»

Falls für die gleiche Methode mehrere Attributprofil notwendig werden, wird der Name mit dem Prozessstatus erweitert.

«Modul»_«Objektklasse»_«Methode»_«Prozessstatus»

Bspw:

Object.Create	SPS_Spesenanhang_Create
Object.CreateAutoImport	SPS_Spesenanhang_CreateAutoImport
Object.Distribute	SPS_Spesenantrag_Distribute_Zusatzprüfung

Hinweise

- Wird ein Attributprofil mehrfach verwendet, behält es den Namen der Erstverwendung.
- Standard Attributprofile erhalten als Modul-Kurzbezeichnung
  HE: health engine
  BE: business engine
  PE: public engine

  IE: generell i-engine (Standard Attributprofile)
- Zu jeder Methode gibt es ein Methodenprofil. Dessen Bezeichnung entspricht dem Methodenname. bspw. «Object_Create».
- Einige SQL-Typen werden im Zusammenhang mit Attributen eingesetzt. In diesem Fall enthält der SQL-Bezeichner auch einen Bezug auf das betreffende Attributprofil. Da Oracle ein Einschränkung der Bezeichnerlänge kennt, muss ggf. eine Kurzbezeichnung (Mnemonic) für das Attributprofil festgelegt werden. Der Mnemonic wird im Beschreibungsfeld des Attributprofils festgehalten, sollte max. 12 Zeichen lang und innerhalb des Moduls eindeutig sein. Eine ggf. angefügte Beschreibung wird über « - » vom Mnemonic abgetrennt.

Standard Attributprofile

Die Erstellung und Verwaltung von Attributprofilen ist ein Kostenfaktor. Die Anzahl eingesetzter Attributprofile muss deshalb minimiert, und falls möglich, sind bereits vorhandene einzusetzen.

Methodenprofile MP

- Zu jeder Methode ist ein Methodenprofil verfügbar, welches die Attribute im Read/Write Zugriff anzeigt
- Das MP ist immer die bevorzugte Wahl, wenn kein Profil benötigt wird, oder keine speziellen Anforderungen bestimmt sind und die Felder schreibar sein dürfen.

Standard Attributprofile

- IE_Default_fix: Dieses Profil zeigt die üblichen Objektfelder im Readonly Zugriff.
- Für die Methoden *Object.Commit()* stehen einige Varianten zur Verfügung, deren Einsatzbereich aus der Bezeichnung hervorgehen:
  IE_Default_Commit
  IE_Default_Commit_Verfolgen
  IE_Default_Commit_Notiz

Attribute bearbeiten

DB Action

Die Maskenfelder werden direkt in der Tabelle ATTRIBUTE_SHOW bearbeitet.

Feld	Beschreibung
SEQ	Reihenfolge
SHOW_TYPE	Die Anzeigemaske setzt sich aus verschiedenen Teilbereichen zusammen. Diese Teilbereiche werden als «Show-Type» bezeichnet. Für jeden Teilbereich werden die Felder mit ihren Parametern definiert. s. *«Hinweise zu Show-Typen»* (Seite 133)
V_POS H_POS	Vertikale Position und Horizontale Position Die Anzeigefläche wird in ein Raster mit definierbaren Dimensionen eingeteilt. Jedes Attribut nimmt eine bestimmte Position ein.
DISPLAY_NAME .._L1,.._L2,.._L3	Textlabel des Attributes
DE- FAULT_VALUE .._L1,.._L2,.._L3	Vorbesetzter Wert, wird aber nur beim Erzeugen eines neuen Eintrages angewendet – ausser bei temporären Feldern. «?clear»: leert den Inhalt des Feldes «?today»: heutiges Datum (ohne Zeit) «?now»: aktuelles Datum und Zeit Zeichenfolge: wird übernommen Datum für Datumsfeld muss im Format «yyyy-mm-dd» definiert werden, in einem Textfeld muss es in der gewünschten Form vorgegeben werden. «?nnn»: SQL-Statement-ID
ATTRIBUTE_ NAME	Datenbankfelder Zu jedem Objekt stehen folgende Attributfelder zur Verfügung: ■ 40 Zeichenfelder à 100 Zeichen -> sys_attribute.t1 .. t40 ■ 10 Zeichenfelder à 2000 Zeichen -> sys_attribute_long1 .. 10 ■ 10 Datumsfelder -> sys_attribute.d1 .. d10 ■ 20 Ganzzahlenfelder -> sys_Attribute.n1 .. n20 ■ 10 Dezimalfelder -> sys_attribute.f1 .. f10

Feld	Beschreibung
	■ beliebige Anzahl Imagefelder -> sys_image.0 .. n.value Prinzipiell können alle Datenbankfelder verwendet werden, die für das aktuelle Objekt definiert sind. z.B: sys_object.descr, sys_object.objname oder sys_object.obj_index Es können auch temporäre Felder, die nicht abgespeichert werden, eingesetzt werden. -> temp.xxxx
TAG	Art des Feldes «FINDER»: Sucher «POPUP»: Pop-up Fenster «BUTTON_..»: Button «DATE»: Datumsfeld «TEXT»: Textfeld «TEXT_ML»: mehrsprachiges Textfeld «TEXTAREA»: Textbereich «PASSWORD»: Passwort Eingabe «RADIO»: Optionstaste «CHECKBOX»: Auswahlkästchen «SELECT»: Dropdown-Liste «HIDDEN»: temporär nicht angezeigt «MULTICHOSEN»: Resultatfeld bei Mehrfachauswahl «SINGLECHOSEN»: Resultatfeld bei Auswahl «EMPTY»: leere Zelle im Anzeigeraster «FIXTEXT»: nur Lesetext «TITLE»: nur Lesetext, aber speziell ausgezeichnet «MAIL»: E-Mail Inhalt, nur für spezielle Methode verfügbar «FILE»: lokale Dateiauswahl, nur für spezielle Methode verfügbar. «SELECTMASTER»/«SELECTSLAVE»: Abhängige Felder «IMAGE»: Bild s. auch Abschnitt «*Hinweise zu Show-Typen* (Seite 133)»
MANDATORY	Zwingende Eingabe: «JA»: Eine Eingabe ist erforderlich «NEIN»: Eingabe möglich «FIX»: keine Eingabe möglich
CHECKER	Client Checker: ein clientseitiger Mechanismus übernimmt die Überprüfung der Feldeingabe. Für mögliche Eingabewerte s. Abschnitt «*Daten-Checker* (Seite 145)»
DIS-PLAY_NAME_	Länge des Labels in %

Feld	Beschreibung
WIDTH	
TAG_PREFS	s. Abschnitt «*Eingabe Validierung* (Seite 145)»
COLSPAN ROWSPAN	Anzahl Spalten und Anzahl Zeilen Das Feld kann mehrere Positionen im Raster belegen. Entsprechend sind die Anzahl Kolonnen und Reihen anzugeben.
PAGE_NR	Attribut Seite Bei der Verwendung mehrerer Attributseiten: Auswahl der zu verwendenden. s. Hinweise.
SERVER_ CHECKER	SQL-Befehlsstring, welcher die serverseitige Überprüfung der Eingabe vornimmt. Nur ein Eintrag für alle Felder des Formulars. s. Abschnitt zu «Server-Checker»
POPUP_TYPE	Nicht verwendet
POPUP_OBJ_ID	s. Abschnitt «Pop-up Fenster definieren». Bei «FINDER»: Objekt-ID des Suchers Bei «POPUP»: Objekt-ID des Wurzelelementes Bei «SELECT» und «HIDDEN»: Es kann auch die ID eines SQL-Statements angegeben werden, welches den Feldwert liefert.
POPUP_ MANDATORY	Die Feldeingabe kann nur über eine Pop-up Fenster stattfinden; dh. keine direkte Eingabe möglich.
REFERENCE	Wird im Zusammenhang mit Auswahl-/Resultatfelder benötigt. s. Abschnitt «Hinweise zu Show-Typen»
VALUE_ STATEMENT_ID	SQL-Statement-ID welches den Rückgabewert umrechnen
FORMAT	Festlegen des Ausgabeformates für das Feld Beispiele: Dezimalzahl, 1 Stelle nach Komma: «format=number grouping=YES frac=1» Datum und Zeit; «format='dd.mm.yyyy hh24:mi'» Verwenden eines CSS Styles: colorbgbar1bg
POPUP_P1	Methode der Baumanzeige

Feld	Beschreibung
POPUP_ PROFILE_ID	Attributprofil für Anzeigen im rechten Datenfenster
POPUP_ DIALOGPREFS	Aufsetzen des Festerlayouts bei Pop-up Fenster

Hinweise

- Die Kombination aus SHOW_TYPE, ATTRIBUTE_ID und SEQ muss eindeutig sein.
- Zu jedem Objekt steht ein kompletter Satz von Attributfeldern zur Verfügung. Sollten diese nicht genügen, können weitere Attributsets zugeordnet werden. Zusätzliche Attributseiten werden im Vorlageobjekt der Objektklasse zugewiesen. Als Konsequenz muss jedem Attribut die zu verwendende Attributseite im Feld *Attribut Seite* zugewiesen werden.
- Hinweise zu Konfigurierung der Eingabefelder finden Sie in den Abschnitten
  «Pop-up Fenster definieren»
  «Eingabeelemente»
- Wird bei einem Feld mit Tag «HIDDEN» die Position ausserhalb des sichtbaren Bereichs gesetzt, wird das Feld für die Berechnung der Layoutgrösse ignoriert und hat keinen Einfluss auf die Darstellung. Ansonsten wird das Feld als Lücke im Layout angezeigt – analog zum Tag «EMPTY».
- Das Feld *Default Wert* wird bei jedem Neuladen der Maske aufgesetzt - unter folgender Voraussetzung
  - Im Feld *Value Statement* ist eine SQL-Statement ID.
  - Dieses SQL-Statement liefert 2 Werte zurück: Anzeigewert | Datenwert
  - Für das Feld wird eine Variable *temp.xxxx* verwendet.
- Den Tag «BUTTON_..» gibt es in folgenden Varianten – wobei diese den entsprechenden HTML-Tags entsprechen: BUTTON_SUBMIT, BUTTON_HREF, BUTTON_RESET, BUTTON_CLOSE sowie BUTTON_EDITFRAME.
  Für die beiden Fälle BUTTON_SUBMIT und BUTTON_HREF enthält das Feld *Tag Prefs* den zugehörigen Methoden-ID.
- Der Tag BUTTON_SCRIPT erlaubt die Ausführung von speziellen Java-Scripts, um z.B. Feldkalkulationen oder Abhängigkeiten durchzuführen. Der Java-Script ist im Feld *Tag Prefs* aufgeführt. Allgemein verfügbare Scripts sind u.a. save(), saveAndClose() oder reload().
  Einschränkung: Da nur der HTML Client dieses Tag unterstützt, kann es für neue Anwendungen nicht mehr verwendet werden.
- Der Tag IMAGE erlaubt die Anzeige eines Bildes auf der Dialogmaske. Im WinClient kann das Feld MANDATORY auf «JA» oder «NEIN» gesetzt werden. Dadurch besteht die Möglichkeit, über das Kontextmenü eine Dateiauswahlmaske aufzurufen und den Bildinhalt auszutauschen. Im Feld TAG_PREFS wird die Anzeigegrösse auf der Maske definiert; Bsp:

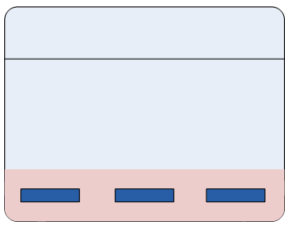
«height=64 width=64». Die ursprüngliche Grafik muss ggf. auf diese Grösse umgerechnet werden. Die Umrechnung wird im Feld FORMAT spezifiziert; Bsp. «maxScale=64 64». Bei Bedarf stehen weitere Funktionalitäten wie Drehen der Grafik zur Verfügung.

Hinweise zu Show-Typen

Es sind die folgenden Show-Typen vorgesehen:

HEADER	Daten im Kopf des Formulars – nur zur Anzeige
BODY	Erfassungs- und Anzeigefelder
CONFIRMATI-ON	Separates Anzeigefenster zur Eingabebestätigung Bemerkung: Wird nur bei HTML Client verwendet.
BODY0..n	Separates vorgeschaltetes Auswahlfenster bei Create- und Import-Methoden; oder Registerblätter. s. Abschnitt «*Maske für neue Objekte* (Seite 140)»
TAB	Register Definition s. Abschnitt «*Registerblätter* (Seite 135)»
XML	InfoPath Formular Definitionen s. Abschnitt «*InfoPath* (Seite 229)»
BUTTON	Formular-Schaltflächen s. Abschnitt «*Formular Schaltflächen* (Seite 133)»
HDR-FILE	Übergabedaten an Drittanwendungen s. Abschnitt «*Daten für Drittanwendungen* (Seite 137)»
HIDDEN	Versteckte Datenfelder s. Abschnitt «*Verborgene Felder* (Seite 137)»
CHOSEN	Auswahlfeld dessen Resultat in das Feld «CHOOSEN» geht.
CHOOSEN	Resultatfeld, welches die Auswahlen der Felder «CHOSEN» erhält. s. Abschnitt «*Mehrere Auswahlfelder* (Seite 138)»
TREE_xxxx	Spezialeinträge für individuellen Anzeigetext im Strukturfens-ter. s. «*Strukturanzeige modifizieren* (Seite 155)»

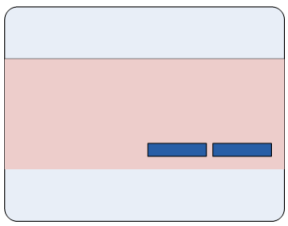
BUTTON – Formular Schaltflächen



Auch der Buttonbereich wird durch das Default-Attributprofil der Methode aufge-setzt. Das Button-Angebot kann durch eigene Show-Type Einträge vom Typ «BUTTON» neu definiert werden. Die verfügbaren Funktionen, welche bei den Schaltflächen hinterlegt werden können, sind durch die verwendete Methode gegeben. Das Default-Attributprofil zeigt, welche Funktionen unterstützt werden.

Feld	**Bemerkung**
Display Name	Beschriftung der Schaltfläche

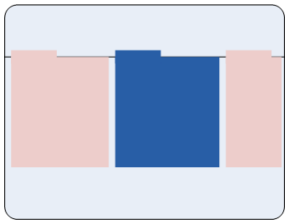
Feld	Bemerkung
Seq Horizontal Position	Position der Schaltfläche von links mit «1» beginnend
Vertical Position	Fix «1»
Mandatory	Bei einem Button-Eintrag muss das Auswahlkästchen gewählt werden. Diese Schaltfläche reagiert als Default auf die Eingabetaste.
Display Name Width	Fix «0»
Tag	Die Art des Buttons: «BUTTON_SCRIPT» «BUTTON_RESET»: Resettaste «BUTTON_EMPTY»: «BUTTON_CLOSE»: «BUTTON_SUBMIT_TO_EXECUTOR»: anderes Objekt «BUTTON_SUBMIT_EDIT_FRAME»: «BUTTON_SUBMIT_EDIT_FRAME_TO_PARENT»:
Tag Prefs	Die Eingaben in diesem Feld sind abhängig von der Art des Buttons. Tipp: Am besten wird das Default-Attributprofil zu Rate gezogen.



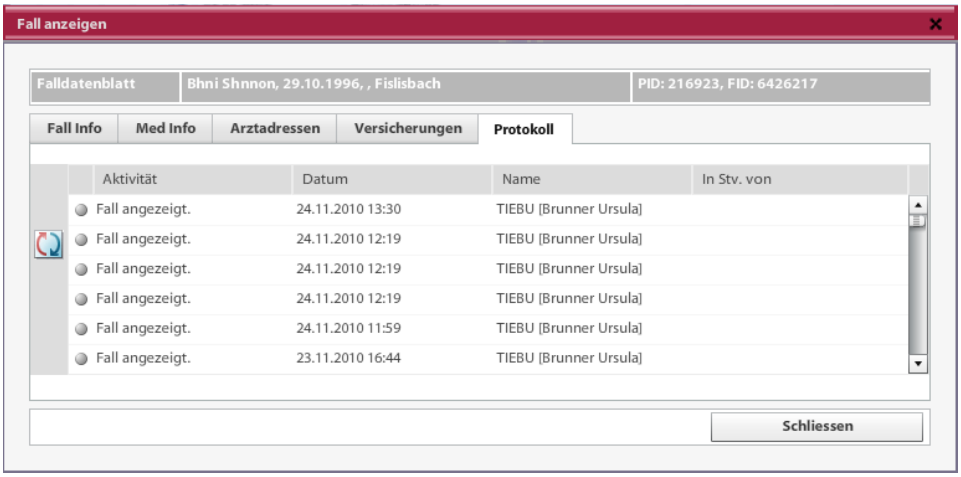
Die im unteren Maskenbereich angezeigten Schaltflächen beziehen sich immer auf das ganze Formular. Bei Masken mit mehreren Registern besteht die Anforderung nach Schaltflächen, deren Aktion sich nur auf die Daten des angezeigten Registers beziehen. z.B. das Speichern von Eingaben.

Um dies zu erreichen, werden die Daten eines «BUTTON» Eintrages nicht als eigener Show-Type «BUTTON» angelegt, sondern im normalen «BODY» des entsprechenden Registers. Der Attributname ist mit «BUTTON» zu belegen.

TAB – Registerblätter



Die Datenanzeige und –erfassung kann auf mehrere Register verteilt werden. Die Maske kann mehrere individuelle Register und drei Standardregister umfassen.



Im Attributprofil werden nur die Anzeigen der individuellen Register definiert. Der Aufbau der Register «Protokoll», «Notizen» und «Wächter» ist vorgegeben und kann nicht geändert werden.

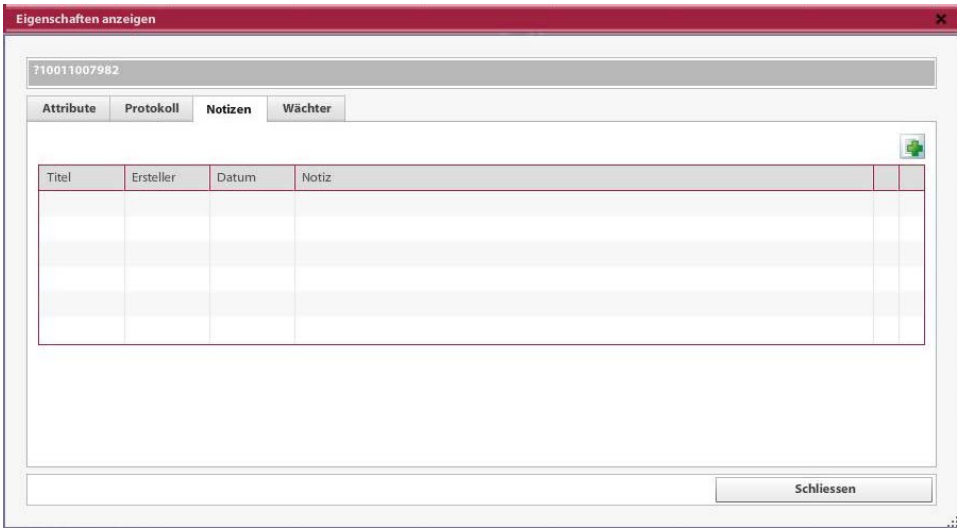
Jedes Register entspricht einem eigenen Show-Typ, dessen Bezeichnung beliebig gewählt werden kann. Üblicherweise wählt man «BODY1», «BODY2» etc. – beachten Sie, dass «BODY0» eine reservierte Bezeichnung ist. Wenn keine Register verwendet werden, wird als Bezeichnung «BODY» verwendet.

Register werden unter dem Show-Type «TAB» angelegt. Im neuen Show-Typ erstellen Sie nun ein neues Attribut für die Register-Parameter.

Feld	Bemerkung
Display Name	Bezeichnung der Registerlasche
Sequenz	Die Anzeigereihenfolge wird durch die aufsteigende Nummerierung der «TAB» Einträge bestimmt.
Tag	«TAB_SUBMIT»: Beim Anklicken der Registerlasche wird eine Funktion ausgelöst.
Tag_Prefs	Funktion, die beim Anklicken ausgelöst wird. Beispiel: Anzeige der Datei /method/object_attributes_view_files/ object_attributes_view0.jsp?show_type=**BODY2**

Feld	Bemerkung
	Für die drei Standardregister sind folgende Einstellungen notwendig: PROTOKOLL: /method/jsplib72/workflow_show.jsp?txnid NOTIZ: /method/jsplib72/notes_edit.jsp?txnid WÄCHTER: /method/watcher_edit.jsp?txnid Es kann auch der Methoden-ID angegeben werden.

Standardregister: Protokoll - Notizen - Wächter



Ein neues Register bedeutet, dass ein neuer Show-Typ etabliert werden muss.

Hinweise

- Der Name des Show-Types muss mit dem verwendeten Parameter im Feld *Tag Prefs* des Register-Eintrages übereinstimmen.
- Der Aufrufparameter «?txnid» ist ein Sicherheitscode, der die Ausführung der Transaktion durch unauthorisierte Aufrufe verhindert (Hackerschutz).
- Die Schaltflächen **Rückgängig** und **Speichern** beziehen sich immer nur auf das aktuell angezeigte Registerblatt.
- Um die Anzeige der Register benutzer-gesteuert anzuzeigen, sind unterschiedliche Attributprofile notwendig. Diese werden den Benutzer über unterschiedliche Aktivitäten angeboten.

HDR-File – Daten für Drittanwendungen

Auch für den Datenaustausch mit Drittanwendungen werden die Datenfelder im Attributprofil festgelegt.

Feld	Bemerkung
Display Name	Feldbezeichnung – für Drittanwendung Wenn in mehreren Prozessstatus der gleiche Datenaustausch möglich sein soll, müssen in jedem Attributprofil identische Namen verwendet werden.
Attribut Name	Datenbankfeld
Seq	Eindeutige Feldreihenfolge
PopupObj ID	Falls nicht der aktuelle Tabellenwert übermittelt werden soll, kann auch ein SQL-Statement den Wert liefern.

Hinweis

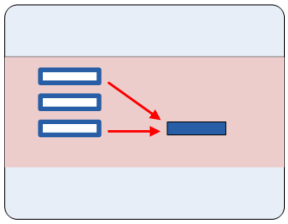
- Im Abschnitt «*Variablen für MS-Office* (Seite 177)» gibt es nähere Angaben zu diesem Anwendungsfall.

HIDDEN – verborgene Felder

Verborgene Felder werden im Show-Type «HIDDEN» angelegt. Diese Felder eignen sich für fixe Sucher- oder Constraint-Parameter, für Umrechnungen oder Verknüpfen von Strings. Ansonsten erfolgt die Definition wie für normale Eingabefelder.

Die Werte verborgener Felder werden in die Datenbank abgespeichert.

CHOSE / CHOOSEN – mehrere Auswahlfelder



Für Realisierung von Auswahlfeldern deren Auswahl in einem Resultatfeld gesammelt werden soll, gibt es zwei Lösungen.

Eigene Maske

Die Auswahlfelder und das Resultatfeld werden in einem separaten Maskenteil angelegt. Dazu stehen 2 spezielle Show-Typen zur Verfügung. Diese Variante kann nur im Zusammenhang mit speziellen Methoden eingesetzt werden: *role.adduser()*, *user.addrole(), object.changeOwner(), object.classify(), object.setShow(), object.setAccess()*.

In einer Maske können mehrere Felder für die Auswahl eines Wertes - «CHOSE» - eingesetzt werden. Das Ergebnis der Auswahlen wird in einem weiteren Feld - «CHOOSEN» - abgelegt. Die notwendigen Felder werden als entsprechende Show-Typen erzeugt.

Show-Type	«CHOOSE»
Display Name	Labeltext
Attribute Name	«temp.xxx» Werte werden in eine temporär Variable abgelegt.
Vertical Position Horizontal Position	Position des Auswahlfeldes
Restliche Felder	Entsprechend der Eingabeart: Sucher, Eingabe, Pop-up,…

Show-Type	«CHOSEN»
Display Name	Labeltext
Attribute Name	Datenbankfeld
Vertical Position Horizontal Position	Position des Auswahlfeldes
Tag	«SINGLECHOSEN»: nur 1 Eintrag «MULTICHOSEN»: mehrere Einträge möglich
Mandatory	«NO»: nicht zwingend «YES»: zwingende Auswahl erforderlich Bei «MULTICHOSEN»: «FIX»: Es wird kein Papierkorb angezeigt, dh. ausgewählte Einträge können nicht mehr gelöscht werden.

Integriert in Maske

Oft sind die Auswahlfelder nur ein Teil der Datenmaske. In diesem Fall werden die Auswahl- und Resultatfelder im betreffenden Show-Typ definiert. Bei der Konfiguration der Einträge ist folgendes zu berücksichtigen:

- Die Zuordnung von Auswahlfeldern und Resultatfeld wird über eine einheitliche Bezeichnung im Feld *Reference* festgelegt.
- Im Resultatfeld ist *Mandatory* auf «YES» und *Tag Prefs* auf «disabled» zu setzen. Damit wird erreicht, dass ein Wert aus einem Auswahlfeld eingeschrieben, aber nicht durch den Benutzer verändert werden kann.
- Im Resultatfeld ist *Tag* auf «MULTICHOSEN» resp. «SINGLECHOSEN» zu setzen. Mehrfachselektion muss natürlich von der Methode unterstützt werden.
- s. auch Abschnitt «*Resultatfeld über Auswahlliste füllen* (Seite 291)».

Maske für neue Objekte

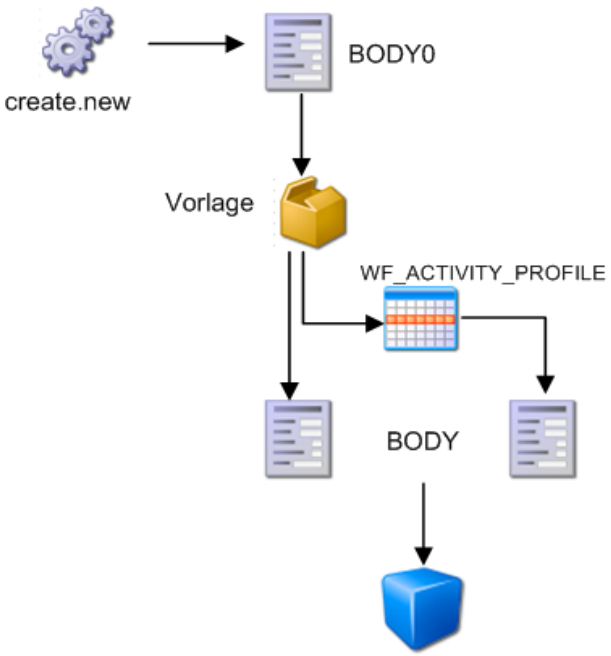
Beim Erzeugen eines Objektes möchte man ggf. in einem ersten Schritt die gewünschte Objektklasse auswählen und in einem Folgeschritt die notwendigen Daten erfassen. Dieser Ablauf kann über ein entsprechendes Attributprofil implementiert werden.

Für die Methode *object.create()* werden keine Eingabemasken mit mehreren Registern unterstützt. Dh. Ein Attributprofil enthält nur Einträge «BODY» – ausser es soll noch ein Auswahl für die Vorlage vorgeschaltet werden.

BODY0 Abschnitt

Falls das Attributprofil Einträge «BODY0» enthält, werden diese Felddefinitionen für die Anzeige einer separaten Masken zur vorgängigen Auswahl einer Vorlage verwendet. Enthält das Attributprofil keinen »BODY0» Eintrag, wird stets die Vorlagendefinition aus der Methodenmaske der Aktivität verwendet.

- Üblicherweise erlaubt die Auswahlmaske die Selektion der gewünschten Vorlage über einen oder mehrere Sucher, Popups oder Listauswahl.
- Die Objekt-ID der Vorlage muss immer in einem Attribut mit der Bezeichnung «template-id» abgelegt werden
- Für die eigentliche Datenerfassung muss das Profil Einträge vom Typ «BODY» enthalten. Diese werden in einer separaten Maske angezeigt. Abhängig von der vorgängig ausgewählten Klasse kann im 2ten Schritt natürlich auch ein alternatives Attributprofil zum Einsatz kommen. Dazu wird in der Tabelle WF_ACTIVITY_PROFIL für die betreffende Aktivität ein Eintrag erzeugt, der auf das alternative Attributprofil verweist - Dieses Verfahren ist für alle Methoden einsetzbar.



Hinweis

- Bei automatisierten Import-Aktivitäten wird die Objektklasse fest vorgeben – via Attributprofil in der Aktivität. Das betreffende Attributprofil enthält folglich keinen «BODY0» Abschnitt. In der Methodenmaske wird zudem *Menu* auf «NONE» gesetzt. Dadurch sind die Berechtigungen aufgesetzt, aber die Aktivität ist für den Benutzer nicht sichtbar.
- Wird in einem Ordner mit gemischtem Inhalt die Methode *object.Create()* mit einer aufgesetzten *Next Activity* für die Bearbeitung verwendet, kann es Fehlermeldungen geben, wenn nicht für alle Objektklassen ein Bearbeitungsprogramm vorliegt – z.B. Word-Dateien gemischt mit Grafikdateien. Idealerweise wird deshalb die Auswahl der Vorlagen in der Auswahlmaske auf die unterstützten Klassen beschränkt.

Tipps & Tricks

Fensterdimensionen

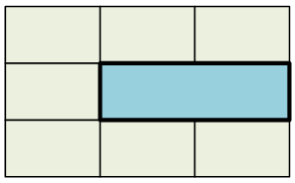
In der Aktivität kann in den *Dialogpräferenzen* eine fixe Fenstergrösse vorgegeben werden oder in *AP Grössenanpassung* kann die dynamische Bestimmung der Grösse angewählt werden. Beim Verwenden der dynamischen Varianten sind folgende Punkte zu beachten:

- Ziel ist die Anzeige der Inhalte ohne unnötiges Scrollen.
- Die Fenstergrösse wird auf den Inhalt der ersten Anzeige ausgelegt. Diese Grösse bleibt unverändert, auch wenn andere Inhalte angezeigt werden.
- Bei Fenstern mit Registern ist es oft unmöglich, den längsten Inhalt immer auf das erste Register zulegen. In solchen Situationen ist auf den Folgeregistern unnötiges Scrollen der Inhalte notwendig.
- Bei Abläufen mit Vormasken – Auswahl einer Vorlage – bleibt die Grösse auch für das Folgefenster erhalten. Die Auswahlmaske ist meist wesentlich kleiner als die eigentliche Eingabemaske.
- Die automatische Grössenberechnung beeinflusst nur die Höhe der Maske.

Die dynamische Grössenberechnung kann nicht verwendet werden bei:

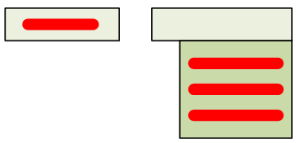
- Masken mit Registern
- Methoden mit Vormasken
- Die minimal erforderliche Maskenhöhe ist 560px

Leere Positionen im Anzeigeraster



Um ein grafisch befriedigende Anzeige zu erreichen, kann es notwendig werden, die nicht besetzten Positionen im Anzeigeraster «schöner» anzuzeigen. Mit Feldern von Typ «EMPTY» kann bei Elementen die über mehrere Anzeigeposition reichen die Anzeige des Rahmens unterdrückt werden.

Abhängige Eingabefelder



Es erleichtert die Auswahl von Werten wesentlich, wenn die Auswahl in einem ersten Feld die angezeigten Werte in einem zweiten Feld einschränkt. z.B. die Auswahl des Kantons schränkt die Angebotenen Gemeinden in einem zweiten Feld ein.

Bei der Konfiguration wird das erste Auswahlfeld im *Tag* mit «SELECTMASTER» markiert. Das abhängige Feld ist im *Tag* mit «SELECTSLAVE» ge-

kennzeichnet. Dabei können auch mehrere Felder vom gleichen «Master» Feld abhängig sein.

In der Maske können gleichzeitig mehrere Abhängigkeiten verwendet werden. Deshalb müssen die Zuordnungen mit Feld *Tag Prefs* festgehalten werden:

- Master Feld: «master=1»
- in allen abhängigen Slave Feldern: «slave=1»

Die Zahl ist entsprechend durchzunummerieren.

Das SQL-Statement für das «Slave» Feld muss mindestens folgende Rückgabewert liefern:

Wert 1	Objekt-ID
Wert 2	Objekt Bezeichnung
Wert 3	Zugehöriger ID des Master-Objektes

Ein Eintrag kann auch für mehrere Master Objekt-IDs zulässig sein. In diesem Fall muss das SQL-Statement je einen Eintrag für die zulässigen Master-IDs liefern.

Schreibschutz

Für die reine Anzeige von Werten innerhalb einer Bearbeitungsmaske gibt es eine Reihe Möglichkeiten.

- Tag=«FIXTEXT»: der Wert aus dem Feld *Default Value* wird angezeigt und kann nicht verändert werden.
- Tag=«TITLE»: auch hier wird der Wert aus dem Feld *Default Value* angezeigt und kann nicht verändert werden. Der Text wird in Fettschrift angezeigt.
- Tag Prefs=«disabled»: Der Feldinhalt kann nicht durch den Benutzer direkt verändert werden.
- Mandatory=«FIX»: Wert kann nicht verändert werden, auch nicht über Auswahlfelder.
- Tag=«SELECT»: In diesem Fall wird das Feld automatisch auf «disabled» gesetzt, so dass die Eingabe nur über die Dropdown Auswahl möglich ist.
- Tag=«FINDER» oder «POPUP»: Fall nur eine Auswahl über den Sucher resp. Pop-up möglich sein sollte, sind folgende Einstellungen notwendig:
  Mandatory=«YES» oder «NO», je nach Erfordernis
  Popup Mandatory=«YES», Auswahl über Finder/Popup
  Tag Prefs=«disabled», Schreibschutz für Feld

Kombination von Auswahl und manueller Eingabe

Es ist nicht möglich in Resultatfeldern neben der Übernahme der Werte von Auswahlwerten auch eine manuelle Eingabe zuzulassen. Eine Umwandlung einer Texteingabe in die notwendige Objekt-ID ist nicht möglich.

MULTICHOSEN Felder

Die Höhe eines MULTICHOSEN Feldes wird im Attributprofil mit «Anzahl Zeilen = 1» angegeben. Die gewünschte Anzahl Zeilen für dieses Feld wird in der Tabelle OBJECT_PROPERTY definiert.

OBJ_ID	??
SEQ	«0»
KEY / VALUE	«SELECT_SIZE» Anzahl Zeilen angeben

Es ist möglich die minimale resp. maximale Anzahl Einträge in der Liste anzugeben. Dies wird ebenfalls über Einträge in der Tabelle OBJECT_PROPERTY vorgenommen:

OBJ_ID	??
SEQ	«0»
KEY / VALUE	«MIN_CARDINALITY» Minimale Anzahl Einträge, damit Formular abgeschlossen werden kann.
	«MAX_CARDINALITY» Maximale Anzahl Einträge, damit Formular abgeschlossen werden kann.

Eingabe Validierung

Die Überprüfung von Benutzereingabe kann prinzipiell

- serverseitig – über Server-Checker

oder

- clientseitig – über Daten-Checker

erfolgen.

Daten-Checker

Die Validierung von Feldeingaben wird durch sog. Daten-Checker durchgeführt. Für übliche Standardfälle steht eine Reihe von vorbereiteten Prozeduren (Javascript, ActionScript, ausprogrammiert) zur Verfügung. Folgende Daten-Checker sind im Standard vorhanden:

Checker	Beschreibung
checkObjID	Nicht verwendet
checkObjName	Gültiger Objektname
isNumber	Nummerischer Wert
isCurrency	Nummerischer Wert mit Nachkommastellen
checkPasswords	Check1: entspricht Passwort den Richtlinien Check2: Überprüfen ob zwei Felder identisch sind Parameter in *Tag Prefs*: Originalfeld: min=x max=yy id="pass1" Zweitfeld: id="pass2" Die Richtlinien von checkPasswords und für Oracle Profiles müssen auf einander abgestimmt sein.
checkContains	Auflösen von Spezialzeichen/Joker für Volltextsuche
checkUser	Firmentypische Überprüfung des Benutzernamens
checkDate	Datumswert
checkLength	Minimale und maximale Länge prüfen Parameter in *Tag Prefs*: min=x max=yy Es kann auch nur der Maxwert angegeben werden.
checkMail	E-Mailadresse, dh. «@» und «.» sind vorhanden
checkDirName	Verzeichnisname

Checker	Beschreibung
checkWatcher	Nicht verwendet

Neuen Daten-Checker definieren

Die Funktionen für HTML Client sind in der Datei «../jslib/checker/checker.js» abgelegt. Anwendungstypische Erweiterungen können in dieser Datei ergänzt werden. Jeder Daten-Checker hat zudem einen Eintrag in der Tabelle P_CODE.

Feld	Bedeutung
TAB_NAME	Alle Einträge müssen den Wert «CHECKERS» haben.
SEQ	Reihenfolge der Einträge in Pop-up Menü, wenn die Daten-Checker im Attributprofil ausgewählt werden können.
INTERNAL_VALUE	Name der Java Funktion. Achtung: Schreibweise und Gross/Kleinschreibung muss identisch wie im Java-Code sein.
DISPLAY_VALUE_ SHORT und LONG sowie L1, L2 , L3	Bezeichnung des Checkers. Wird im Auswahl Pop-up Menü angezeigt. Es besteht die Möglichkeit zwei unterschiedliche Längen zu definieren. Welcher konkret verwendet wird, hängt natürlich vom SQL-Befehl ab. Für alle Sprachen gleich aufsetzen
DESCR	zusätzliche Beschreibung; wird nicht verwendet

Wie wird die Überprüfung durchgeführt?

Wenn im Attributprofil ein Daten-Checker für ein Feld ausgewählt wird, wird der Wert von INTERNAL_VALUE als zusätzliches HTML-Tag im Eingabefeld mitgegeben.

Der eigentliche Code für die Überprüfung des Feldinhaltes ist in der Datei \jslib\checker\checker.js enthalten. Für das Überprüfen des Feldinhaltes wird die Funktion «verify» mit einer Referenz auf das aktuelle Eingabeobjekt aufgerufen.

Die Funktion testet, welche Checker Tags im Eingabefeld vorhanden sind, und löst eine entsprechende Überprüfungsroutine aus. Da das Feldobjekt übergeben wird, hat der Algorithmus Zugriff auf alle vorhandenen DHTML-Komponenten des Eingabefeldes.

Parameter für die Überprüfung mitgeben

Im Attributprofil können im Feld TAG PREFS können weitere Tag-Felder angeben werden, die im Prüfalgorithmus ausgewertet werden können.

z.B. min=10 max=100

Die beiden Werte können berücksichtigt und als Grenzwerte eine Eingabe verwendet werden. Bei der Bezeichnung solcher zusätzlicher Tags ist man frei, solange man keine reservierten Bezeichnungen von DHTML verwendet.

Server-Checker

DB Action

Der Server-Checker wird im Attributprofil hinterlegt und überprüft die Daten, wenn das Formular an den Server zurückgeschickt wird. Diese Überprüfung eignet sich für Felder, die gegenüber der Datenbank geprüft werden müssen.

Der Server-Checker wird als PL/SQL-Prozedur implementiert. In der Tabelle ATTRIBUTE_SHOW im Feld SERVER_CHECKER wird direkt der SQL-Befehl abgelegt. Der SQL-Befehl basiert üblicherweise auf einer PL/SQL-Prozedur (C_SC_xxx), die folgenden Rückgabewert gibt:

Status	«0»: ok «2»: Fehler
Text	Fehlermeldung

Der Rückgabewert ist ein String, der beide Werte getrennt mit einem Leerzeichen enthält.

Hinweise

- Es wird nur ein einziger Server-Checker pro Attributprofil definiert, um Leistungsproblemen vorzubeugen. Dieser Server-Checker kontrolliert alle Feldeingaben.
- Prinzipiell ist es egal, in welchem Feldeintrag der Server-Checker eingetragen wird. Es sollte aber ein überprüftes Feld sein.
- Bei Registerwechsel wird die Maske vom Server neu geschickt. Deshalb wird bei einem Registerwechsel immer auch der Server-Checker ausgelöst.
- Der Server-Checker wird direkt als PL/SQL-Prozedur implementiert.
- Der Server-Checker kann für alle Attributprofile eingesetzt werden, die Datenveränderungen zulassen.
- Parameter-Übergabe: Parameter mit «?{xxx}» - z.B. «?{object.obj_id}» oder «?{activity.activity_id}» - werden zu Laufzeit ausgewertet. Parameter ohne «?{}» werden als Konstanten weitergegeben, wobei der Typ anhand der Funktionsdefinition ermittelt wird. Strings müssen ohne «'» resp. «"» angegeben werden. Alle Maskenparameter müssen in Grossschrift angeben werden: Bsp. «?SYS_ATTRIBUTE.N12}» d.h. in Grossschrift, obwohl im Attributprofil in Kleinschrift definiert.
- Falls der Server-Checker ungültige Werte findet, werden diese nicht abgespeichert. Falls es die Anwendung erfordert, kann dieses Verhalten übersteuert, und die fehlerhaften Daten trotzdem abgespeichert werden. Dazu

muss in der Tabelle OBJECT_PROPERTY ein Eintrag vorgenommen werden:
OBJ_ID: ID der Aktivität
KEY: «SAVE_BEFORE_SERVER_CHECKER»
SEQ: «0»
VALUE: «YES»

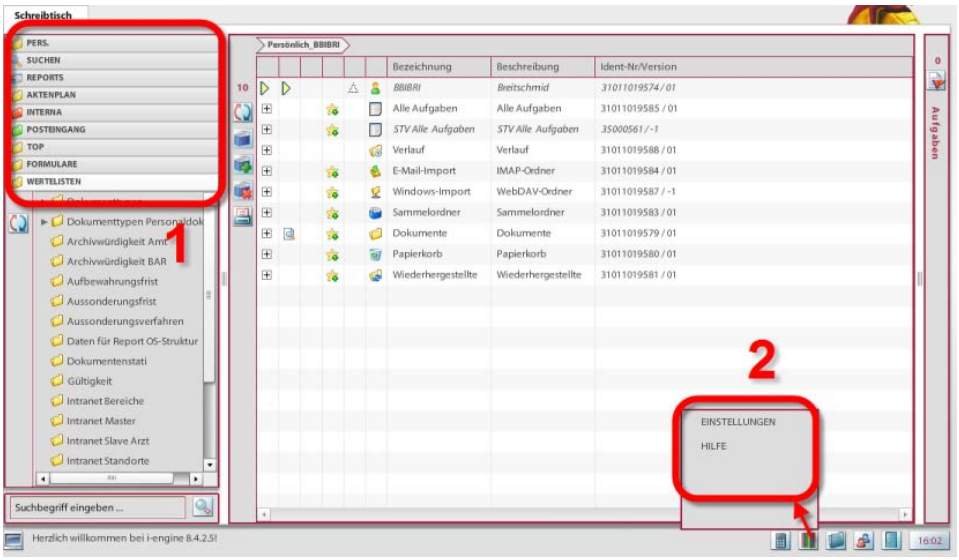
Achtung: Der Prozess verbleibt aber im aktuellen Status, da ja die Bedingung für ein Weiterführen nicht gegeben ist.

- Statt den Server-Checker im Attributprofil zu hinterlegen, kann er auch an die AKtivität gekoppelt werden. Dazu wird ein entsprechender OBJECT_PROPERTY Eintrag notwendig:
OBJ_ID: ID der Aktivität
KEY: «SERVER_CHECKER»
SEQ: «0»
VALUE: ID des SQL-Statements

Menüleiste & Strukturanzeige

DB Action

Über die Menüleiste stehen dem Benutzer bei Anzeigestelle 1 und Anzeigestelle 2 je 10 Funktionen direkt zur Verfügung. Das Customizing erfolgt direkt in der Datenbank. Die Menüleiste ist für jeden Anwender individuell.



Prinzipiell kann jedes beliebige Objekt über einen Menüeintrag als Startobjekt zur Anzeige im Strukturfenster gebracht werden. Üblicherweise werden die Ausgaben der Funktionen aus der Anzeigestelle 2 in einem separaten Fenster und nicht im Strukturfenster angezeigt. Hinweis: In der betreffenden Aktivität wird dazu das Feld *Anzeige (Hauptfenster)* mit «_blank» aufgesetzt.

Es muss eine Zuordnung zwischen Menüpunkt und Startobjekt definiert werden. Dies geschieht in der Tabelle PROPERTIES. Die Einstellungen müssen direkt in dieser Tabelle eingetragen werden. Jeder der 2x10 Menüpositionen hat normalerweise 3 Einträge.

USER_NAME	Die Einstellungen der Menüleiste sind für jeden Benutzer individuell aufgesetzt.
KEY	Für jeden Menüpunkt können folgende Records eingetragen werden: «MENU_OBJ_ID»: Startobjekt für Anzeige im Strukturfenster «MENU_DESCR»: Menübezeichnung

	«MENU_URL»: aufzurufende URL «TARGET_DIR»: Temporär Verzeichnis
SEQ	Reihenfolge der Menüeinträge Anzeigestelle 1: 0..9 Anzeigestelle 2: 10..19
VALUE L1/L2/L2	Alle Einträge können sprachabhängig angelegt werden.

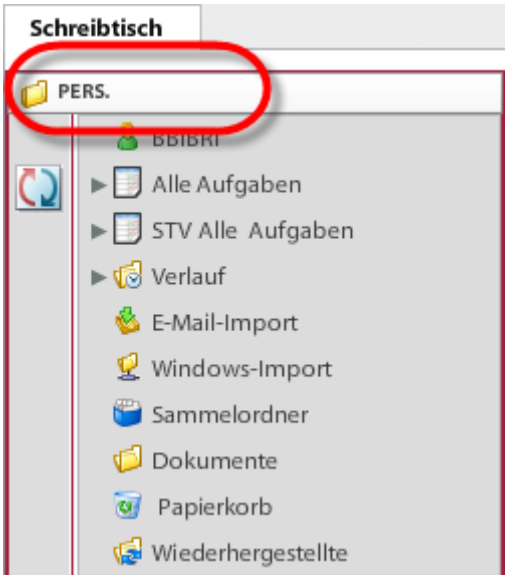
Hinweis

- Meistens werden die administrativen Funktionen auf die Anzeigestelle 2 gelegt.

Konfigurieren der Menüleiste

Startobjekt

Über einen Menüpunkt wird immer eine Aktion ausgelöst, welche das Strukturfenster mit Inhalt füllt. Mit dem Startobjekt wird das Grundelement der Baumstruktur festgelegt. Es wird empfohlen, den Startpunkt auf ein Objekt mit Objektklassentyp «Root» zu legen, um den Benutzer nicht mit Pfadangaben zu erstaunen, die über das Startobjekt hinaus zurückgehen.



Auch dieses Objekt befindet sich in einem definierten Prozessstatus, aus welchem ggf. Aktivitäten aufgerufen werden können. s. weiter unten «Objektliste im Datenfenster anzeigen»

Üblicherweise ändern sich die Status der Hauptordner nicht. Deshalb werden die Status aller Startobjekte zentral in einem Pseudoprozess gesammelt.

Aufzurufende URL

Prinzipiell gibt es zwei Aktionsarten, welche der Menüpunkt im Strukturfenster auslösen kann:

Baumanzeige	Über eine JSP-Funktion wird der Baum ab dem Startobjekt aufgebaut und angezeigt. Es stehen 2 Varianten zur Verfügung, welche sich im Bedienungsverhalten und der Performance unterscheiden. «/method/object_tree.jsp»: Der gesamte Baum wird beim Aufruf aufgebaut. Dies kann zu langen Antwortzeiten beim Anklicken des Menüpunktes führen. Bei der Navigation im Baum bleibt die Struktur jeweils erhalten (wie beim Windows Explorer). Hinweis Diese Methode sollte sehr restriktiv eingesetzt werden. «/method/object_tree2.jsp»: Beim Aufruf wird nur der aktuelle Ast des Baumes bestimmt. Dadurch eignet diese Anzeige auch für sehr grosse Strukturen. Allerdings wird immer nur ein Teilbaum angezeigt. Hinweis Beim Einsatz der zweiten Methode wird die Baumanzeige bei der Umstellung auf einen Stellvertreter nicht neu aufgesetzt. Dies führt dazu, dass der Stellvertreter Aktivitäten sieht, die er nicht auslösen kann, oder Funktionen vermisst. Bei Baumanzeigen, die von solchen Umstellungen betroffen sind, sollte deshalb die ersten Methode eingesetzt werden.
Dialogmaske	In diesem Fall wird über den Menüpunkt direkt eine Dialogmaske im Strukturfenster angezeigt. Meist ist dies die Eingabemaske eines Suchers. «/method/mrb.jsp?obj_id=x&activity_id=y&parent_id=z»: Das Bestimmen des notwendigen Eintrages kann wie folgt erfolgen: ■ In der Anwendung eine Seite mit einer Verknüpfung auf den gewünschten Sucher anzeigen. ■ Quelltext der Seiten anzeigen und den «methode/mrb.jsp» Eintrag für den betreffenden Link suchen.

	■ Eintrag kopieren, aber ohne «&wname» und folgende Parameter Hinweis: Maskenfelder können auch bereits vorbesetzt angezeigt werden. Dies kann mittels zusätzlicher Parameter beim URL-Aufruf bewerkstelligt werden: «..&TEMP.OBJ_NAME=xxx»

Temporärverzeichnis

Dateien, welche vom Server an das Clientsystem übertragen werden, sind auf dem Temporärverzeichnis zwischengespeichert. Standardmässig wird dieser Eintrag einmalig, für alle Benutzer identisch, aufgesetzt. Dazu wird ein Record mit dem USERNAME=«SYSTEM» eingetragen. Falls notwendig können aber auch benutzertypische Einträge vorgenommen werden.

Es ist möglich Systemvariablen einzusetzen.

Beispiele:

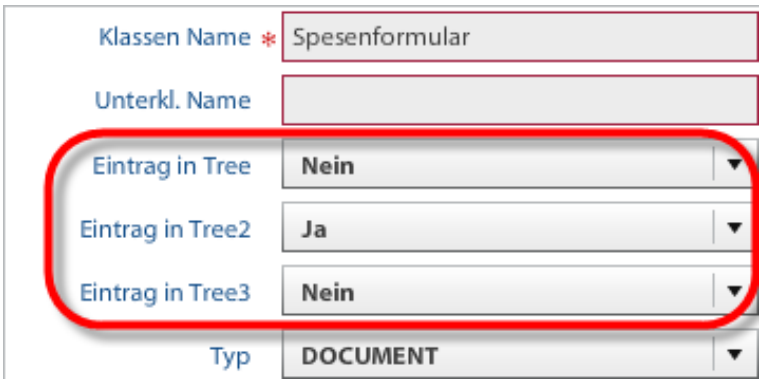
«%LOCAL_APPDATA_PDCS%»

«%MYDOCUMENTS%\e-DMS»

Was wird im Baum angezeigt?

DB Action

Es lässt sich steuern, welche Objekte in der Baumanzeige erscheinen. In der Objektklasse wird der Defaultwert entsprechend aufgesetzt.



Beim Erzeugen eines neuen Objektes, wird dieser Wert übernommen und bestimmt, ob das neue Objekt im Baum angezeigt wird. Das Anzeigeverhalten kann aber für individuelle Objekte auch abgeändert werden.

Die Einstellungen werden direkt in der Tabelle STRUCTURE vorgenommen. Für jedes Objekt, das in der Baumanzeige erscheint, findet sich ein Eintrag in dieser Tabelle.

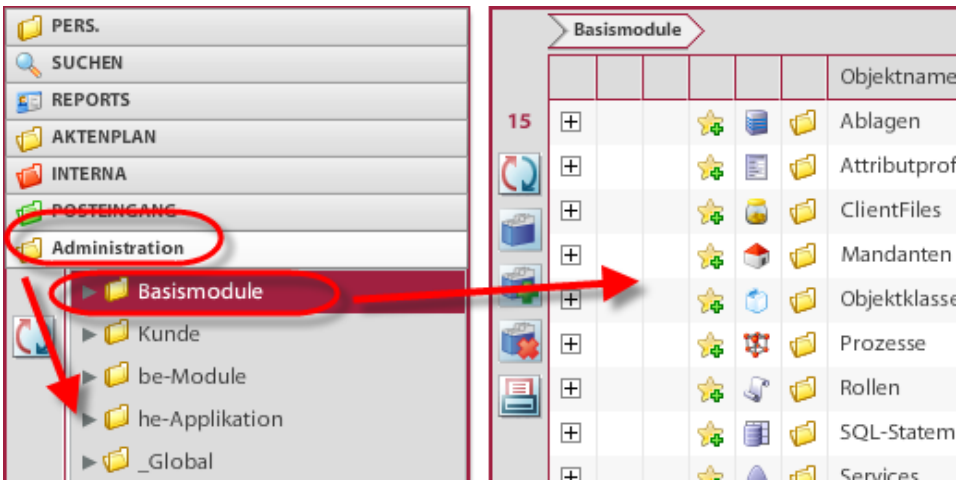
TYPE	Für diese Art Einträge fix ORIGINAL
UPPER	Startobjekt, entsprechend der Baumanzeige.
LOWER	Objekt-ID des Objektes, das in der Baumanzeige aufgeführt wird
TREE TREE2 TREE3	Die automatische Auflistung des Objektes kann separat für Hauptfenster, Projektfenster und Popup festgelegt werden. «YES»: anzeigen «NO»: nicht anzeigen

Objekte im Datenfenster anzeigen

Zusammen mit der Anzeige des Strukturbaumes kann auch die Anzeige im rechten Datenfenster nachgeführt werden. Für das Startobjekt muss dazu im aktuellen Status eine Aktivität (Methode *object.objlist())* definiert sein, welche bei Anzeige im Tree ausgelöst wird (*Behavior*=«YES»).

Beispiel

Über das Menü *Administration* wird die Baumanzeige mit dem Startobjekt «Administration» aufgerufen. Ist keine Startaktivität - d.h. das betreffende Feld *Verhalten* ist eingeschalten - für das Startobjekt festgelegt, wird die Anzeige im Datenfenster erst nachgeführt, wenn ein Element der Strukturanzeige angewählt wird. Dieses Verhalten ist oft eher verwirrlich und sollte deshalb nicht verwendet werden.



1. Menüeintrag

Tabelle PROPERTIES

USER_NAME	KEY	VALUE	SEQ
BBIBRI	MENU_DESCR	TOP	6
BBIBRI	MENU_OBJ_ID	10000000	6
BBIBRI	MENU_URL	/method/object_tree2.jsp	6

2. Startobjekt

Das Objekt «Management» (Object-ID: 10011453805) hat den Prozess «ManagementProcess» zugewiesen. Dieser befindet sich im Prozesszustand *ManagementStatus*.

3. Anzeigen im Baum

Die Child-Objekte des Startobjektes sind von der Objektklasse «Ordner». Bei dieser Klasse ist der Default auf «Eintrag in Tree=JA» gesetzt. Beim Erzeugen der 6 Ordner wurde dieser Wert übernommen. Deshalb sind in der STRUCTURE Tabelle im Feld TREE alle auf «YES» gesetzt – und werden somit angezeigt.

Tabelle STRUCTURE

UPPER	LOWER	TREE	TYPE
10011453805	10011455521	YES	ORIGINAL
10011453805	10011455585	YES	ORIGINAL
10011453805	10011455635	YES	ORIGINAL
10011453805	10011453806	YES	ORIGINAL
10011453805	10011453948	YES	ORIGINAL
10011453805	10011453949	YES	ORIGINAL

4. Anzeige im Datenfenster

Für den Prozesszustand *ManagementStatus* ist eine Aktivität **Öffnen** mit «Behavior=YES» definiert. Damit wird die hinterlegte Methode *object.objList()* ausgeführt, wenn das Startobjekt in der Strukturanzeige angezeigt wird.

Strukturanzeige modifizieren

Standardmässige wird in der Baumanzeige als Anzeigetext der Objektname verwendet. Ähnlich wie die Menüleiste kann jedoch auch der Anzeigetext benutzerabhängig festlegt werden. Dadurch kann der Inhalt des Anzeigetextes den individuellen Bedürfnissen angepasst werden. Bsp. kann die Anzeige mit einem Verantwortlichen oder mit dem letzten Updatedatum erweitert werden.

Die notwendige Zusatzinformation wird in der Tabelle ATTRIBUTE_SHOW hinterlegt. Um des Anzeigetext vom Benutzerprofil abhängig zu gestalten sind folgende Massnahmen notwendig.

1. Anzeigetext definieren

In der Tabelle ATTRIBUTE_SHOW wird der gewünschte String für den Anzeigetext definiert. Für jeden Objektklassentypen, dessen Anzeigetext vom Standard (Objektname) abweichen soll, wird ein Eintrag notwendig:

SHOW_TYPE	TREE_xxxx xxxx entspricht dem Objektklassentypen; Bsp. TREE_PATIENT oder TREE_DOCUMENT

ATTRIBU-
TE_NAME
	SQL-Statement Text
	liefert genau einen Wert, den Anzeigetext zurück
	
	Als Parameter steht ?{object.obj_id) zur Verfügung.
	
	Bsp.
	`SELECT o.obj_name || '-' || c.type`
	`FROM object_tbl o, objclass_tbl c`
	`WHERE o.obj_id=?{object.obj_id} AND`
	`o.objclass_id=c.objclass_id`

2. Menüeintrag anpassen

In der Tabelle PROPERTIES muss für den betreffenden Menüeintrag der «MENU_URL» Wert ergänzt werden:

/methode/object_tree.jsp?attribute_profile_id=nnnnnnn

Der Wert von nnnnnn entspricht dem Attributprofil, wo die Erweiterungen abgelegt wurden.

Hinweis

- Bei einer Methode *Project.Edit()* kann das bereits bestehende Attributprofil der Aktivität mit den zusätzlichen Einträgen erweitert werden.

Objekte – Dokumente & Ordner

i-engine ist ein Objekt-basierendes System. Alles und jedes ist dementsprechend ein «Objekt». Für die Systemkonfiguration sind jedoch zwei Objektarten speziell interessant:

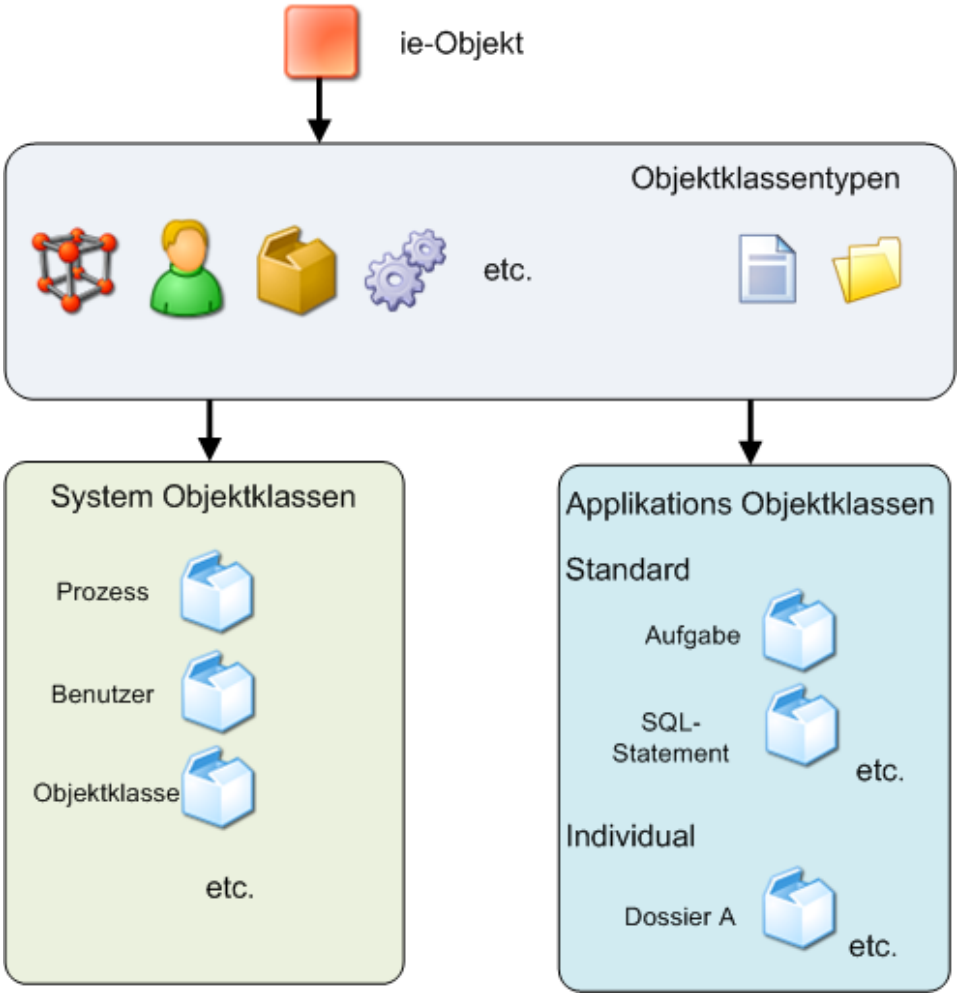
- Objekte für Inhalte (Dokumente, Dateien, Informationen,..) und
- Objekte für die Strukturierung (Ordner)

Durch die Festlegung von Objektklassen können auf einfache Weise Objekte mit gleichen Eigenschaften erzeugt und verwaltet werden.

Objektklassen abgeleitet aus Objektklassentypen

Objektklassentypen

Die Grundelemente der Objekthierarchie in *i-engine* sind die **Objektklassentypen**, welches sich aus dem Wurzelelement «ie-Objekt» ableiten. Der Objektklassentyp bestimmt schlussendlich die Attribute & Methoden eines Objektes und damit die verfügbare Funktionalität im Prozessablauf. Objektklassentypen können weder erweitert noch neue hinzugefügt werden. Im Unterschied dazu können Objektklassen abgeändert und neu erzeugt werden.



System-Objektklassen

Basierend auf den vorgegeben Objektklassentypen von *i-engine* enthält das System für jeden Objektklassentypen eine entsprechende Objektklasse. Im Normalfall ist eine Erweiterung dieser System Objektklassen nicht notwendig, ausser in folgenden Fällen:

- «FINDER» und «CONSTRAINT»: wenn unterschiedliche Symbole angezeigt werden sollen.

- «CLIENTFILES»: für jede zu unterstützende Dateierweiterung ist eine Objektklasse notwendig.

Applikations-Objektklassen

Üblicherweise sind Objekte in den Geschäftsprozessen aus Applikations-Objektklassen abgeleitet. Im Rahmen der Business Process Analyse (BPA) werden die erforderlichen individuellen Objektklassen festgelegt.

- Applikations-Objektklassen haben ausschliesslich folgende Objektklassentypen: «DOCUMENT», «OBJLIST»; Healthcare: «PATIENT», «STUDY», «SERIE», «VISIT»
- *i-engine* enthält eine Vielzahl Objektklassen bereits im Standardumfang. Sie finden diese Klassen im Verzeichnis «Basismodule > Objektklassen» So z.B.
  - Aufgabe
  - Benutzer
  - Mitarbeiter-Rolle
  - Ordner
  - Papierkorb
  - SQL-Statement
  - Sucher
- Für den Kunden werden auch individuelle Objektklassen angelegt. Die individuellen Applikations-Objektlassen werden unter «Kunde > Stammdaten > Objektklassen» abgelegt.
- Ein einfacher Grund für die Notwendigkeit neue Objektklassen zu erstellen, sind angepasste Symbole oder spezielle Objektbezeichnungen.

Aufstellung Objektklassentypen

Der Anwendungszweck der Objektklassentypen ist durch *i-engine* vorgegeben. Teilweise sind Objekte bestimmter Objektklassentypen die Voraussetzung, um überhaupt das Funktionen von *i-engine* zu ermöglichen.

	ACTIVITY Aktivitäten im Geschäftsprozess
	ALIAS Verknüpfungen auf Objekte
	APPOINTMENT Exchange Anbindung: entspricht einem Termineintrag der über das IMAP-Protokoll vom Exchange-Server importiert wurde. Wird in «AppointmentList» abgelegt.
	APPOINTMENTLIST Standard Element im «Persönlich» Baum: **Termine** Jeder Benutzer hat ein Objekt dieses Typs für die Ablage der Termine.
	ATTRIBUTE_PROFILE In den Attributprofilen sind die Angaben für den Aufbau der Eingabe- und Anzeigemasken enthalten.
	CLIENTFILE Zusatzdateien, die beim Anmelden mit AutoUpdate auf die Workstation geladen werden sollen.
	CONSTRAINT Listenelemente, die einen gefilterten Inhalt auflisten. Die notwendigen Suchparameter sind bereits aufgesetzt.
	DOCUMENT Basisobjekt; Wird für alle Dokument-artigen Objekte ein-gesetzt.
	FINDER Listenelement für Sucher, wobei vorgängig eine Maske zur Eingabe der Suchparameter angezeigt wird.
	HISTORY Standard Element im «Persönlich» Baum: **Verlauf** Jeder Benutzer hat ein Objekt dieses Typs für die Ablage der Verlaufseinträge.
	HISTORY_ENTRY Verlaufeintrag – bei einem Klick wird nicht das Verlaufob-

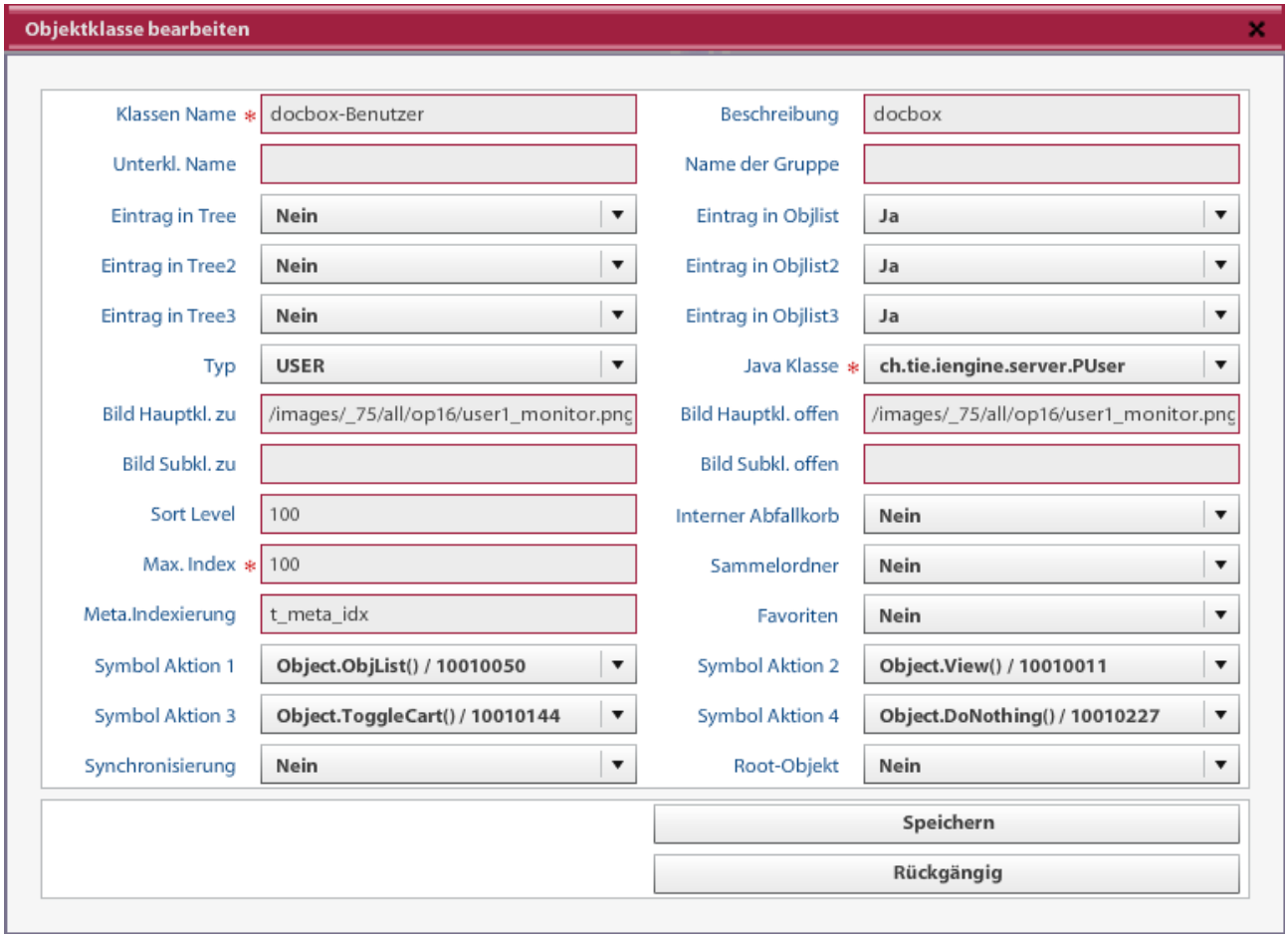
	jekt sondern direkt das betroffene Objekt geöffnet.
	HISTORY_FOLDER Standard Element im «Persönlich» Baum: **Alle** Jeder Benutzer hat ein Objekt dieses Typs für die Ablage der Einträge.
	IMAP Standard Element im «Persönlich» Baum: **E-Mail Import** IMAP Definitionen: Festlegen der Verbindungsparameter für den Import über das IMAP Protokoll
	INSTANCE Objekt des Standard Patientendossiers. Entspricht bspw. den DICOM Bildern.
	iTAN Persönliche iTAN-Codes, welche beim Anmelden kontrolliert werden.
	LOCATION Physischer Ablageort für Dokumente
	MAIL E-Mail-Meldungen
	OBJCLASS Objektklassen Objekte
	OBJLIST Ordner Elemente, wird aber auch für Prozessstatus Objekte verwendet.
	OU Organisationseinheit, enthält Objekte vom Typ «Role»
	PATIENT Hauptobjekt in der Standard Patientenstruktur
	PROCESS Prozess im Geschäftsprozess
	RECYCLEBIN Standard Element im «Persönlich» Baum: **Papierkorb** Jeder Benutzer hat ein Objekt dieses Typs für die Ablage temporär gelöschter Objekte.
	RECYCLED Standard Element im «Persönlich» Baum: **Wiederhergestellte** Jeder Benutzer hat ein Objekt dieses Typs für die Ablage

	von wiederhergestellten Objekten, deren Ordner zwischenzeitlich gelöscht wurde.
	ROLE Der Rolle werden die Systembenutzer zugewiesen. *i-engine* kennt unterschiedliche Rollenkategorien für Anwendungsfälle wie hierarchische Rollenkonzepte oder Projektteams. Hierarchierollen werden den Organisationseinheiten zugewiesen.
	URL Integration externe Referenzen.
	USER Benutzer
	SERIES Seriedaten innerhalb des Standard Patientendossiers.
	SERVICE Kommunikation mit Drittanwendung via Webdienste oder Message Services.
	SQL-STMT SQL-Statements
	STATUS Prozessstatus im Geschäftsprozess
	STUDY Studien-Objekt innerhalb des Standard Patientendossiers
	STYLESHEET Stylesheet CSS Dateien. Die Dokument-Objekte werden in der Tabelle verwaltet.
	TASK Exchange Anbindung: entspricht einem Aufgabeneintrag der über das IMAP-Protokoll vom Exchange-Server importiert wurde. Wird in «TaskList» abgelegt.
	TASKLIST Standard Element im «Persönlich» Baum: **Alle Aufgaben** Jeder Benutzer hat ein Objekt dieses Typs die Ablage der Aufgaben.
	SHOPPINGCART Jeder Benutzer hat ein Objekt dieses Typs für die Ablage der Speziallinks bei einem *object.Distribute()*. Diese werden für die Kontrolle der Zugriffsberechtigungen (Berech-

	tigungsstufe) für Objekte im Distribute Status benötigt.
	WEBDAV Standard Element im «Persönlich» Baum: **Windows Import.** WebDAV Definitionen: Festlegen der Verbindungsparameter für den Import über das WebDAV Protokoll
	VISIT Fall-Objekt innerhalb des Standard Patientendossiers.

Eine Objektklasse erzeugen

Im Ordner «Objektklassen» stellt das Kontextmenü die Funktion **Objektklasse erzeugen** zur Verfügung. Damit wird die Eingabemaske für die Erstellung einer neuen Objektklasse aufgerufen.



Feld	Beschreibung
Klassen Name	Name des Dokumententyps. Dieser Name muss eindeutig gewählt werden.
Beschreibung	Ergänzende Beschreibung
Unterkl. Name	Name des verbundenen Unterdokumentes. Die Bezeichnung muss nicht eindeutig gewählt werden. Aktuell wird dieses Feature nicht verwendet.
Name der Gruppe	Beliebige Dokumente können unter einer Gruppenbezeichnung zusammengefasst werden. Damit lassen sich Dokumente zusammenführen und durchsuchen, welche z.B. über unterschiedliche Prozesse erstellt werden.

Feld	Beschreibung
	Eine Gruppe kann auch in einem Sucher verwendet werden, um eine Suche über mehrere Objektklassen durchzuführen
Eintrag im Tree	Die Instanzen dieses Dokumentes werden im linken Strukturfenster angezeigt. Die Einstellung ist separat für das (1) Hauptfenster, (2) Projektfenster und (3) Pop-up einstellbar.
Eintrag in ObjList	Die Instanzen dieses Dokumentes werden im rechten Datenfenster angezeigt. Die Einstellung ist separat für das (1) Hauptfenster, (2) Projektfenster und (3) Pop-up einstellbar.
Typ	*i-engine* stellt eine Reihe von «Objektklassentypen» zur Verfügung, aus welchen sich neue Objektklassen ableiten lassen. Der Objektklassentyp definiert schlussendlich das Verhalten des Objektes. Die Objektklassentypen sind im Konzeptteil der Dokumentation erläutert. In der täglichen Customizing-Praxis werden nur die Typen «OBJLIST» und «DOCUMENT» verwendet.
Java Klasse	Die Java-Klasse implementiert das Verhalten für einen bestimmten Objektklassentyp. deshalb besteht zwischen diesem Eintrag und dem Eintrag im Feld *Typ* ein unmittelbarer Zusammenhang. So muss z.B. für eine Objektklasse des Typs «PROCESS» die Java-Klasse auf «ch.tie.iengine.server.PProcess» gesetzt werden. *i-engine* wird mit einem Satz von Basis-Objektklassen ausgeliefert, die selten erweitert werden müssen. In der täglichen Customizing-Praxis wird deshalb nur die Java-Klasse «ch.tie.iengine.server.PObject» benötigt, welche für die Objektklassentypen «OBJLIST» und «DOCUMENT» zuständig ist.
Bild Hauptkl. Zu Bild Hauptkl. Offen	Symboldarstellung für die geschlossenen resp. geöffneten Zustandsanzeigen
Bild Subkl. zu Bild Subkl. Offen	Für die geschlossene resp. geöffnete Zustandsanzeige des verbundenen Unterdokumentes
Sort Level	Festlegen der Anzeigereihenfolge in der Baum- und Listanzeige. Die Objekte werden entsprechend dem *Sort Level* ihrer Objektklasse in aufsteigender Folge angezeigt. Die Sortierung erfolgt alphanumerisch. Alle Objekte in der Baumanzeige und in der Listenanzeige

Feld	Beschreibung
	werden in erster Priorität immer nach dem Sort-Level der Objektklasse sortiert. In der der 2ten Stufe ist die Sortierung in der Baumanzeige immer durch den Objektnamen gegeben. In der Listenanzeige erfolgt die initiale Anzeige in 2ter Stufe nach dem ersten Anzeigefeld. Diese kann interaktiv geändert werden.
Interner Abfallkorb	Das Objekt kann einen eigenen Papierkorb verwenden, in welchem gelöschte Child-Objekte abgelegt werden. Voraussetzung ist, dass ein Objekt des Objektklassentyps «Recyclebin» im Client-Objekt definiert ist. Diese Feature wird aktuell nicht verwendet.
Max Index	Entsprechend der angegeben Zahl erzeugt das System die Versionierungs-Information. Über das Kontextmenü **Indices editieren** können die erzeugten Versionskennzeichnungen anschliessend bearbeitet werden.
Sammelordner	Definiert, ob Objekte in den Sammelordner gelegt werden können. «JA»: Objekte dieser Objektklasse können gewählt werden. «Nein»: Funktion nicht verwenden.
Attrib.Indexierung	Für die Suche können auch die Daten des Attributprofils indexiert werden. «JA»: auch Inhalte der Attribute indexieren «NEIN»: bei der Suche nicht berücksichtigen s. «*Volltextsucher* (Seite 281)»
Favoriten	Legt fest, ob das Favoritensymbol für diese Objekte zur Verfügung steht. Damit kann im Browser ein Lesezeichen für den direkten Zugriff auf das Objekt erstellt werden. «JA»: Symbol anzeigen – Lesezeichen unterstützen «NEIN»: Symbol nicht anzeigen
Symbol Aktion	In der Listendarstellung können zu jedem Eintrag 4 Symbole angezeigt werden, die eine direkte Aktion auf das Objekt auslösen. s. dazu auch den Abschnitt «*Symbol Aktionen* (Seite 170)» Falls eine Aktion gewünscht wird, muss die betreffende Methoden-ID angegeben werden. Entsprechend sollte im aktuellen Prozessstatus eine Aktivität mit dieser Methode vorhanden sein – idealerweise nur eine einzige. Falls keine Ausführungsberechtigung für die betreffende Aktivität besteht, bleibt die Anzeigeposition leer. Wird

Feld	Beschreibung
	überhaupt keine Aktivität auf der betreffenden Position gewünscht, so wird am besten die Methode *Object.doNothing()* zugewiesen.
Synchronisierung	Werden mehrere Mandanten in einem System unterstützt, können die Objekte zwischen diesen synchronisiert werden. «JA»: Alle Objekte dieser Objektklasse synchronisieren «NEIN»: Objekte nicht synchronisieren
Root-Objekt	Objekte als sog. «Root» Elemente in der Hierarchie markieren.

Zugriffsrechte auf neue Objekte erteilen

- Damit der aktuelle Benutzer eine Instanz dieser Objektklasse (ein Objekt) erstellen kann, muss er das Zugriffsrecht auf das Objektklassen-Objekt besitzen. Sie müssen deshalb nach dem Erzeugen einer neuen Objektklasse unbedingt die Rollen für die betreffende Objektklasse korrekt aufsetzen.
- Damit ein neues Objekt vom Benutzer aufgelistet werden kann, muss ihm das Recht **Sichtbarkeit** erteilt worden sein. Da auch diese Eigenschaft vom Systemobjekt der Objektklasse auf die Instanz vererbt wird, muss sie dort entsprechend aufgesetzt werden. Üblicherweise ist die Sichtbarkeit für Objektklassen nicht eingeschränkt, dh. das neue Objekt hat die Objektrechte seines Containers.
- Ein Objekt wird in *i-engine* zudem nur angezeigt, wenn der Benutzer mindestens eine Aktivität im aktuellen Status verwenden kann; d.h. beachten Sie, dass ein neues Objekt auch entsprechende Aktivitäten aufweist, da sonst kein weiterer Zugriff darauf möglich wird.
- Damit ein Objekt sichtbar ist, muss ein Zugriffsrecht auf seine Objektklasse gegeben sein. Damit lassen sich auf einfache Weise ganze Objektgruppen für bestimmte Benutzer sperren.

Hinweis

In der Praxis wird praktisch ausschliesslich der Wert für das Sichtbarkeitsrecht aufgesetzt. Nur bei Objektklassen und Objektvorlagen wird die Möglichkeit, abweichende Zugriffrechte erteilen zu können, ausnahmsweise eingesetzt.

Bildsymbole für die Objektklasse

Damit Bilddateien für die Anzeige von Symbolen verwendet werden können, muss folgendes beachtet werden:

- Der Web-Server muss die Datei über einen virtuellen Zugriffspfad finden. Die virtuellen Verzeichnisse müssen mit dem Administrationsprogramm des Web-Servers erstellt werden.
- Die Symboldatei muss zuerst manuell auf dieses Verzeichnis umkopiert werden.
- Am besten verwenden Sie dazu PNG oder GIF-Dateien. Das Symbol muss in einer Auflösung von 16x16 und 32x32 Bit vorliegen. Die Dateien müssen jeweils im entsprechenden Verzeichnis zur Verfügung gestellt werden; Bsp. «..\ap16» und «..\all\ap32». In den Eigenschaften wird der Pfad auf die 16x16 Version angegeben..
- Es empfiehlt sich, alle Symboldateien auf dem Unterverzeichnis *../images* abzulegen. Der physikalische Speicherort ist von Ihrer Installation abhängig – üblicherweise ist dies auf dem Web-Server-Rechner das Verzeichnis */Oracle/AS/j2ee/demo/applications/iengine/pdcs/images*.

Katalog - Standard Symbole für Objektklassen

	1	2	3	4	5	6
A						
B						
C						
D						
E						
F						
G						
J						
K						
L						
M						
N						
O						

Hinweis:

- Zusätzlich sind weitere Symbol-Bibliotheken verfügbar.

Symbol Aktionen

In der Listendarstellung können bei jedem Eintrag mehrere Symbole angezeigt werden. Sie ermöglichen teils die Durchführung von Aktivitäten, teils dienen sie der Informationsanzeige.

Zusammenstellung der möglichen Symbol-Positionen:

Pos	Bedeutung	Definitionen
1	Symbol Aktionen 1 An erster Stelle wird üblicherweise das Öffnen des Objektes platziert.	Aktivität: Methode in Objektklasse > Symbol Aktion Symbol: Pro Methode in WF_METHODE definiert Tooltip: aus Aktivität > Imperativ
2	Symbol Aktionen 2 An zweiter Stelle wird üblicherweise das Anzeigen des Objektes platziert.	Wie oben
3	Symbol Aktionen 3 An dritter Stelle wird üblicherweise der Sammelordner angezeigt. Aktuelles Objekt in Sammelordner eintragen	Wie oben Symbol: durch Methode gegeben Tooltip: fix
4	Symbol Aktionen 4 An vierter Stelle wird üblicherweise das Favoriten-Symbol angezeigt. Ein Link auf Eintrag wird in Web-Browser Favoriten einge-	Wie oben Symbol: durch Methode gegeben Tooltip: fix

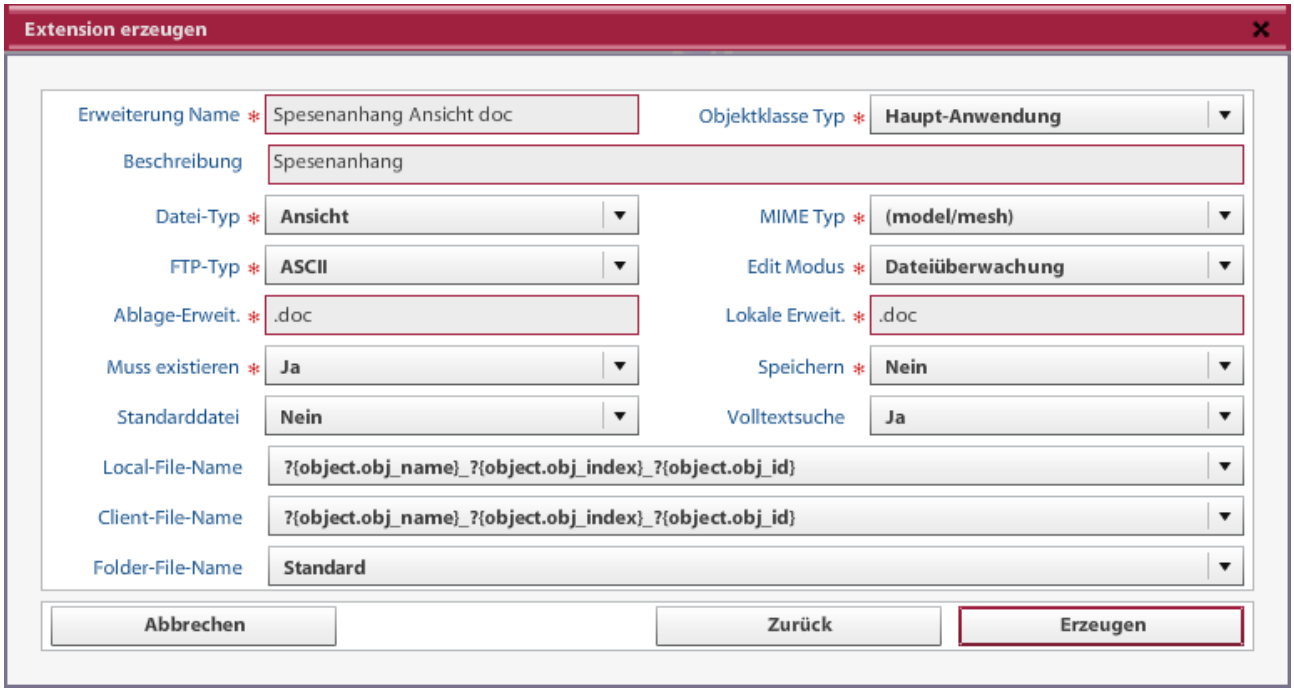
Pos	Bedeutung	Definitionen
	tragen	
5	Prozessstatus – bei Workflow	Aktivität: Kontextmenü anzeigen Symbol: Prozessstatus > Bild Tooltip: Prozessstatus > Statusname ggf. mit Notiz aus Profil erweitert.
6	Objektklasse	Aktivität: Kontextmenü anzeigen Symbol: Objektklasse > Bild Hauptklasse Tooltip: Objektklasse > Klassenname

Objekt Extension

Wenn *i-engine* Objekte ein Dokument repräsentieren, muss ein Zusammenhang zwischen dem Objekt und den zugewiesen Dokumenten festgelegt werden. In der «Extension» wird dem Objekt ein bestimmter Dokumentetyp zugeordnet, welcher sich durch eine eindeutige Dateierweiterung (z.B. .doc, .xls, ppt) identifizieren lässt.

Dabei können auch mehrere Dateierweiterungen der gleichen Objektklasse, oder - mit Einschränkungen - die gleiche Extension für unterschiedliche Objektklassen definiert werden.

In der **Extension** wird neben der Dateierweiterung auch der Verwendungszweck der Dateien festgelegt. Über das Kontextmenü der betreffenden Objektklasse wird Funktion **Extension Erzeugen** ausgelöst.



Feld	Beschreibung
Erweiterung Name	Kennzeichnung der Dateierweiterung und des Verwendungszwecks
Objektklasse-Typ	Wählen Sie, ob dies eine Extension für ein Hauptdokument oder ein verbundenes Unterdokument ist. Aktuell wird nur «Haupt-Anwendung» eingesetzt.
Beschreibung	Ergänzende Beschreibung
Datei-Typ	Je nach Anwendung werden unterschiedliche Dateien für verschiedene Verwendungszwecke benötigt. *i-engine* unterscheidet drei Verwendungszwecke: Siehe Hinweise. «Anwendung»: Diese Datei wird auf der Arbeitsstation für die Bearbeitung durch die Drittanwendung benötigt. Die Methoden wie *Object.CheckIn/Out()*, *Object.Upload()* oder *Object.Download()* verwenden diese Dateien. Sollte die Drittanwendung mehrere Dateien benötigen, müssen auch mehrere Extensions erstellt werden. «Ansicht»: Diese Datei wird für die Anzeige im Browser benötigt; d.h. der Web-Browser kennt dieses Format, da bereits eingebaut, oder die Funktionalität ist durch Zusätze wie KEY-

Feld	Beschreibung
	view erweitert. Die Methode Object.View() verwendet diese Datei. Nur ein Eintrag möglich. «Header»: Wird für die Variablenübergabe in Office-Anwendungen benötigt. Nur ein Eintrag möglich.
MIME-Typ	Für den Transfer via HTTP-Protokoll muss die Datei mit einem entsprechenden MIME-Typ gekennzeichnet werden. Bei *Datei-Typ*=«Header»: immer «text/plain(wt)»
FTP-Typ	Die Übermittlung der Datei zwischen Applikations-Server und Datenserver wird durch den FTP-Service vorgenommen. «ASCII»: Für die Übermittlung reiner ASCII-Dateien wird ein optimierter Transfer angeboten. Allerdings kann diese Art der Übermittlung nur für reine Textdateien verwendet werden und ist damit nur für Spezialfälle geeignet. Vorteil einer ASCII-Übertragung liegt auch in der Kompatibilität beim Transfer zwischen unterschiedlichen Betriebssystemen. «BINARY»: Normalerweise muss der – unwesentlich langsamere – Binary Transfer verwendet werden.
Edit Modus	Bei einigen Methoden wird nach Abschluss der Bearbeitung, die Datei automatisch wieder eingecheckt. Abhängig von der Drittanwendung sind unterschiedliche Tests notwendig, um das Bearbeitungsende sicher bestimmen zu können. *i-engine* unterstützt deshalb verschiedene Überwachungsmodi. «Dateiüberwachung»: Die Datei wird von Drittanwendung während der Bearbeitung gesperrt. «Anwendungsüberwachung»: Das Bearbeitungsende bedeutet auch die Beendigung der Drittanwendung. «Kritische Überwachung»: Spezieller Modus für die PowerPoint Bearbeitung
Ablage-erweiterung	Erweiterung des Dateinamens, welche im Ablageordner verwendet werden soll.
Lokale Erweiterung	Dateierweiterung, welche auf der Arbeitsstation verwendet werden soll. Diese Angabe ist – einfachheitshalber – durch die Drittanwendung vorgegeben, mit welcher die Datei auf der Arbeitsstation bearbeitet wird.

Feld	Beschreibung
Muss existieren	Bei Dateien für die Anwendung gibt es üblicherweise Dateien, welche unbedingt benötigt werden und andere, die z.B. nur temporär angelegt werden. Bsp. bei einer Win-Word Anwendung muss für die .doc Datei diese Option eingeschaltet werden. Sollte nicht gleichzeitig mit der nächsten Option verwendet werden, da «Speichern» damit sowieso ausgeführt wird. Für Datei-Typ «Anzeige» üblicherweise ausgeschaltet.
Speichern	Beim CheckIn der Dateien werden nur jene Dateien auf den Server zurückgeladen, welche hier markiert sind. Alle anderen Dateien werden von *i-engine* – via JavaScript - auf der Arbeitsstation gelöscht. Sollte nicht gleichzeitig mit der vorherigen Option verwendet werden. Für Datei-Typ «Anzeige» üblicherweise ausgeschaltet.
Standarddatei	Wird für den Import von Dateigruppen verwendet, welche zusammengehören (z.B. bei CAD-Anwendung besteht eine Konstruktion neben der eigentlichen Zeichnungsdatei aus weiteren, notwendigen Dateien. Die Hauptdatei – welche üblicherweise an die Anwendung gekoppelt ist – wird als Standarddatei markiert.
Volltextsuche	Festlegen, ob die zugehörigen Dokumentdateien für die Volltextsuche indexiert werden sollen. «JA»: indexieren für die Volltextsuche «NEIN»: Nicht indexieren
Local-Filename	Bezeichnung der Datei auf dem Applikationserver «Standard» «?{object.obj_name}_?{object.obj_index}_ ?{object.obj_id}» «?{object.obj_name}» Standard: «?{object.obj_id}»
Client-Filename	Bezeichnung der Datei auf der Arbeitsstation des Benutzers «Standard» «?{object.obj_name}_?{object.obj_index}_ ?{object.obj_id}» «?{object.obj_name}» Standard: «?{object.obj_id}»

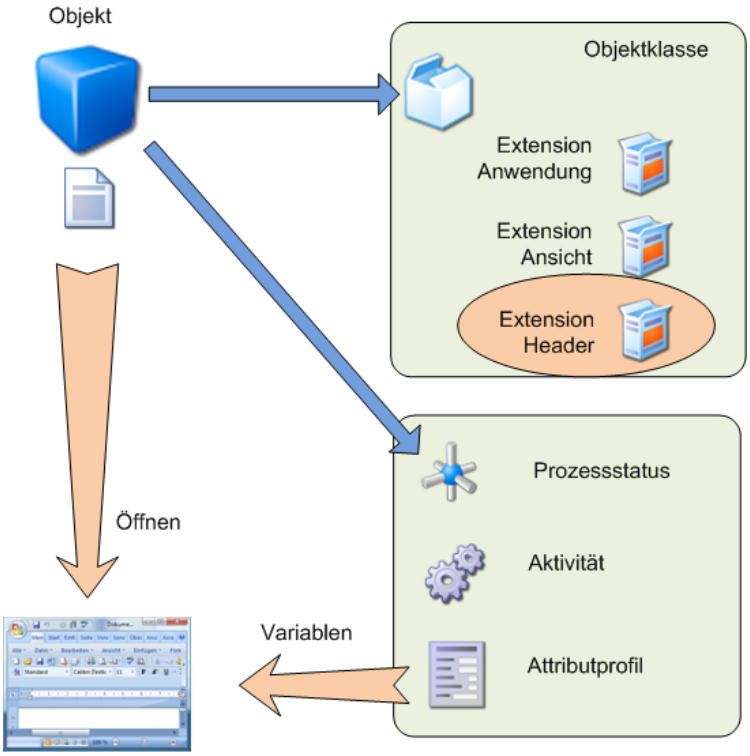
Feld	Beschreibung
Folder-Filename	Bezeichnung der Datei auf der Dateiablage «Standard» «?{object.obj_name}_?{object.obj_index}_ ?{object.obj_id} «?{object.obj_name} Standard: «?{object.obj_id}»

Hinweise

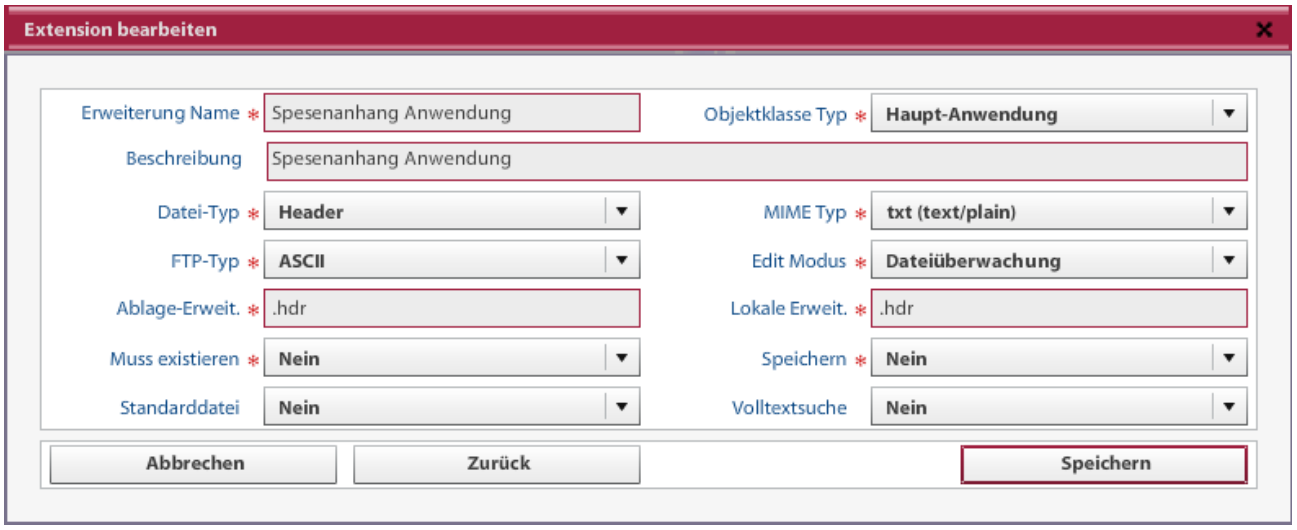
- Auch wenn für Ansicht und Anwendung die gleichen Dateitypen verwendeten werden, sind trotzdem beiden Einträge vorzunehmen.
- Eine Objektklasse kann mehrere, unterschiedliche Objekt-Extensions benötigen, z.B. DOCs und DOTs bei einer WinWord Objektklasse. Während für die «Anwendung» mehrere Einträge möglich sind, ist für «Ansicht» nur eine zulässig.
- Über die Einstellungen bei der Namensvergabe der «Filename» kann sichergestellt werden, dass die Dateibezeichnung auf jeden Fall eindeutig ist.
- Die Definition von «Folder Filename» kann später mit *Extension.Edit()* nicht mehr modifiziert werden.

Variablen für MS-Office Anwendungen

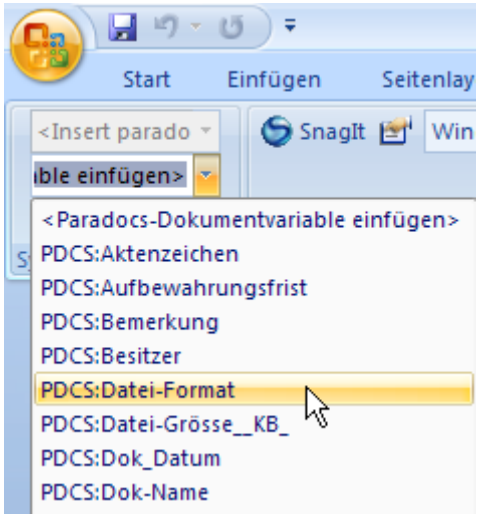
Für MS-Office Anwendungen (WinWord, Excel) können Variablen aus *i-engine* in das betreffende Dokument integriert werden. Die gewünschten Variablenwerte werden über spezielle Einträge im Attributprofil festgelegt.



Zudem muss eine spezielle Erweiterung vom Datei-Typ «Header» erfasst werden. Die Einträge sind vorgegeben.



- Im Attributprofil der betreffenden Aktivität müssen die *i-engine* Variablen festgelegt werden, welche in der MS-Office Anwendung zur Verfügung gestellt werden sollen. Dazu werden Einträge mit dem SHOW_TYPE=«HDR_FILE» erfasst.
- Auf der Arbeitsstation wird die Datei «paradocs_hdr.dot» benötigt. Diese muss durch die Auto-Update Funktion jeweils auf die Arbeitsstation übertragen werden.
- Der Benutzer kann dann in Word die betreffende Variable auswählen und einsetzen.



Hinweis

- Als Variablen können Im Profil die üblichen Attribute wie t_x, n_x, f_x oder d_x verwendet werden. Die Attribute beziehen sich natürlich auf das aktuelle Objekt.
- Es ist auch möglich Variablen an WinWord zu übergeben, welche sich nicht auf das aktuelle Objekt beziehen. In diesem Fall muss in der Tabelle ATTRIBUTE_SHOW im Feld *Popup-ObjID* ein SQL-Statement eingetragen werden, welches den benötigten Wert zurückliefert.

Ablage-, lokale Erweiterung, MIME-Typ

Anhand der Dateierweiterung bestimmt das System den Anwendungszweck der Datei. Daraus ergibt sich:

- Falls unterschiedliche Dateien für die Anwendung oder Anzeige verwendet werden, müssen sie sich in der Ablage-Erweiterung zwingend unterscheiden.
- Auf einem Arbeitsverzeichnis kann eine **Ablage-Erweiterung** nur genau einem Dokumenttyp zugeordnet werden; d.h. nur eine einzige Objektklasse kann eine bestimmte Dateierweiterung pro Verzeichnis benutzen.
- Sie können eine gegenüber der lokalen Erweiterung abweichende Ablagen-Erweiterung definieren. Damit können Sie in einem Ordner den gleichen Dateityp für unterschiedliche Dokumenttypen ablegen. Sie legen z.B. auf einem gemeinsamen Verzeichnis die TIFF-Ausgaben aus einem Elektro- und einem mechanischen CAD ab. Über die unterschiedlichen Dokumenttypen können die zugehörigen Prozesse weiterhin die Dateien den entsprechenden Anwendungen zuordnen.
- Anhand des MIME-Typs bestimmt der Web-Browser auf der Arbeitsstation, wie die übermittelte Datei zu behandeln ist. Moderne Drittanwendungen machen beim Installieren alle notwendigen Register-Einträge, so dass der Web-Browser weiss, wohin er einen bestimmten MIME-Typ weiterschlaufen muss. Der von der Drittanwendung verwendete MIME-Typ muss mit dem im System konfigurierten Wert übereinstimmen. Wenn ein unbekannter MIME verwendet wird, zeigt der Web-Browser einen Dialog an, um die Art der Bearbeitung abzufragen.

Einstellung der Speicheroption und Dateisäuberung

Mit der Auswahl für *Muss Existieren* und *Speichern* wird der Dateitransfer zwischen *i-engine* System und Arbeitsplatz-PC, sowie die Behandlung von temporären Dateien festgelegt.

Es kommen drei Einstellungen für diese beiden Optionen in Frage:

Muss Existieren: «Nein» Speichern: «Nein»	Löschen von nicht mehr benötigten Temporärdateien auf dem Arbeitsplatz
Muss Existieren: «Ja» Speichern: «Nein»	Dokumentdateien, welche auf das *i-engine* System zurückgeholt werden.
Muss Existieren: «Nein» Speichern: «Ja»	Temporärdateien, welche ggf. auf das *i-engine* System zurückgeholt werden.

Häufig benötigen Drittanwendungen eine Reihe von Temporärdateien auf der Arbeitsstation. Mit *i-engine* kann die Behandlung dieser Dateien optimiert werden:

- Wenn die Anwendung auch Temporärdateien erstellt, welche diese aber nicht wieder selbstständig löscht, empfiehlt sich folgende Konfiguration:
- Erzeugen Sie eine Objektextension für diese Datei.
- Schalten sie die beiden Optionen *Muss Existieren* und *Speichern* aus.

Damit wird erreicht, dass diese Datei beim Beenden der Bearbeitung auf der Arbeitsstation automatisch gelöscht wird.

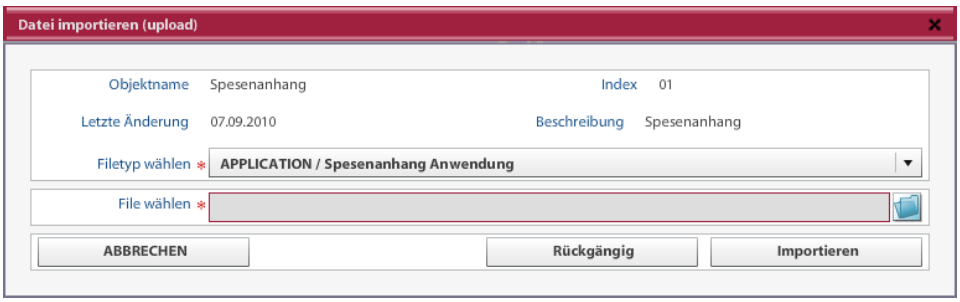
- Wenn die Anwendung Temporärdateien erstellt, welche Zwischenergebnisse enthalten, die bei einer späteren Weiterarbeit wieder verwendet werden können, empfiehlt sich:
- Erzeugen Sie eine Objektextension für diese Datei.
- Die Option *Muss Existieren* wird aus- und *Speichern* eingeschaltet.

Damit wird erreicht, dass eine evtl. vorhandene Datei beim Beenden der Bearbeitung auf der Arbeitsstation ebenfalls abgespeichert wird. Für das Weiterarbeiten wird sie beim *CheckOut* automatisch wieder auf die Arbeitsstation geladen.

Vorlagedateien

Jeder Erweiterung einer Objektklasse kann eine eigene Vorlagendatei zugeordnet werden. Wenn eine Objektklasse z.B. Excel und WinWord Dokumente umfasst, so wird den beiden Erweiterungen je eine eigene Vorlagendatei zugewiesen. Die Vorlagendatei enthält den Inhalt, der für neue Dokumente automatisch aufgesetzt werden soll. Sie wird mit der Drittanwendung erstellt und anschliessend in *i-engine* übernommen.

Über das Kontextmenü der Systemvorlage steht die Funktion **Datei importieren** zur Verfügung.



Im *Filetype* muss festgelegt werden, ob die Datei für «View/Ansicht» oder «Application/Anwendung» verwendet werden soll.

Jeder Objektklasse, welche ein Dokument darstellt, MUSS auch eine zugeordnete Vorlagendatei zugewiesen werden. Andernfalls kann das System davon keine neuen Objekte erzeugen.

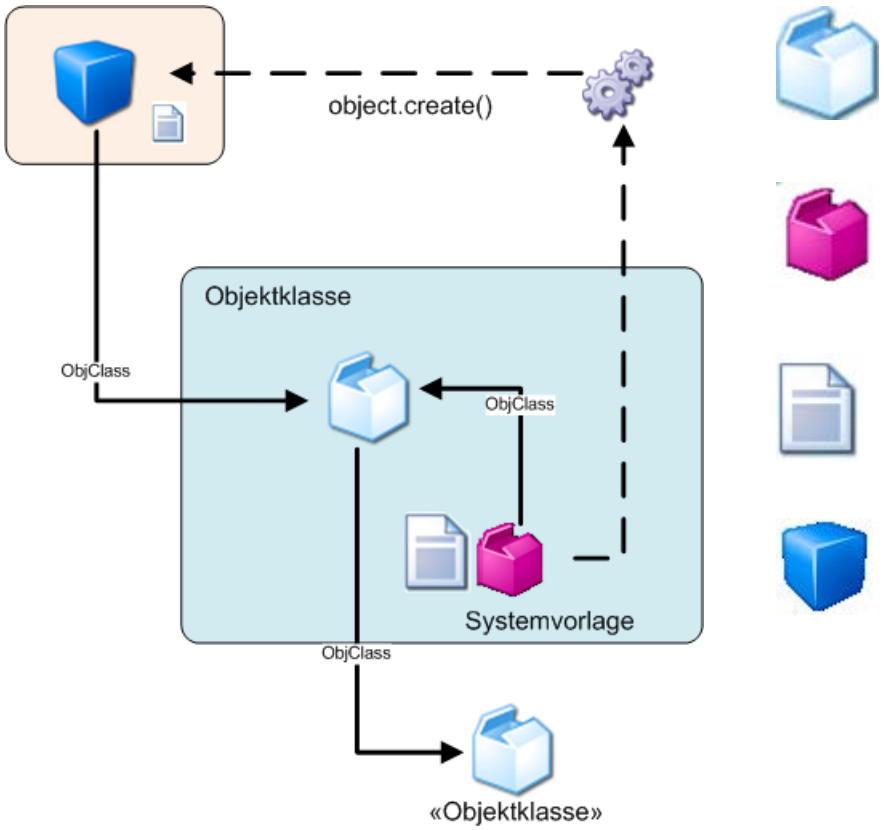
Hinweis

- Bei WinWord gilt zu beachten, dass neue Dateien nicht auf einer «.dot» Datei basieren, sondern dass eine Kopie der «.doc» Vorlagendatei als neue Datei verwendet wird.
- Aktuell kann nicht geprüft werden, welche Vorlagedateien einer Systemvorlage bereits zugeordnet wurden.

Vorlagen für Objekte

Systemvorlage (Objektklasse)

Die einfachste Art ein neues Objekt gemäss einer Vorlage zu erzeugen, ist die Erzeugung einer Instanz der Objektklasse. Das neue Objekt ist in diesem Fall eine Kopie der Systemvorlage. Wenn es sich um ein Dokumentobjekt handelt, wird zudem ebenfalls eine Kopie der Vorlagendatei für das neue Objekt angelegt.



Objekt mit Objektklassentyp «Objektklasse». Repräsentiert die Applikations-Objektklasse «xxx»

Objekt mit Objektklassentyp «xxx». Dieses Objekt ist die Systemvorlage für neue Objekte der Klasse «xxx».

Entsprechende Vorlagendatei(en) falls Extensions für den Objektklassentyp «xxx» definiert sind.

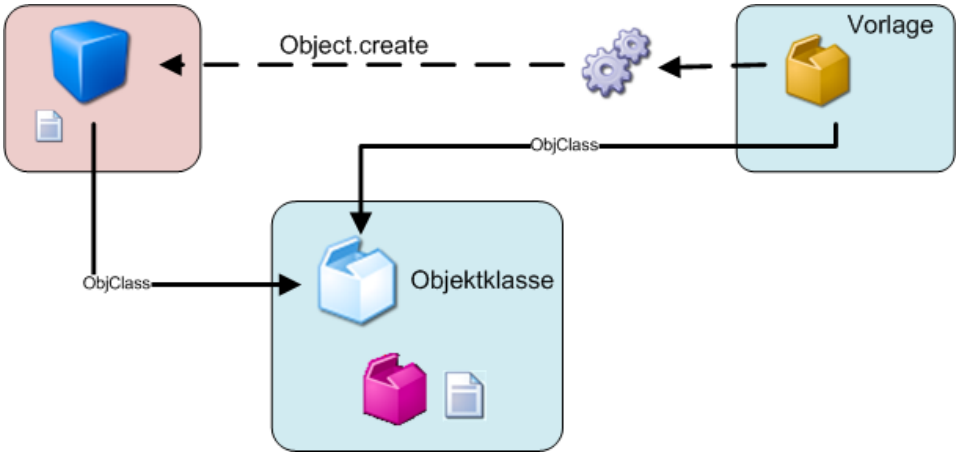
Neues Objekt der Applikationsklasse «xxx»

Die neue Objektinstanz übernimmt sämtliche Einstellungen aus dem «Objektklassen» Objekt. D.h. Werte in den Eigenschaftsfeldern werden entsprechend dem Objekt *Systemvorlage* automatisch aufgesetzt und von den definierten Vorlagendateien werden Kopien für das neue Dokument angelegt.

Vorlagen-Objekt

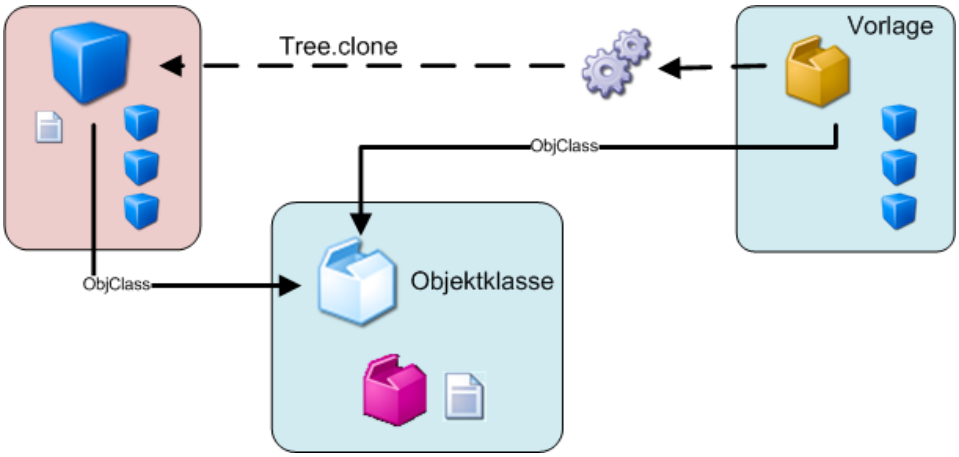
Die Objektklasse des Vorlagen-Objektes bestimmt die Objektklasse des neuen Objektes. Von allen Objekten des Vorlagen-Objektes werden Kopien im neuen Objekt angelegt.

Einfaches Vorlagenobjekt



Strukturiertes Vorlagenobjekt

Neue Objekte können aber auch aus beliebigen abgeleiteten Objekten oder strukturierten Objektgebilden erstellt werden. Dazu wird ein Vorlagen-Objekt angelegt, in welchem die benötigten Objekte angelegt werden. Die Struktur kann beliebig komplex sein.



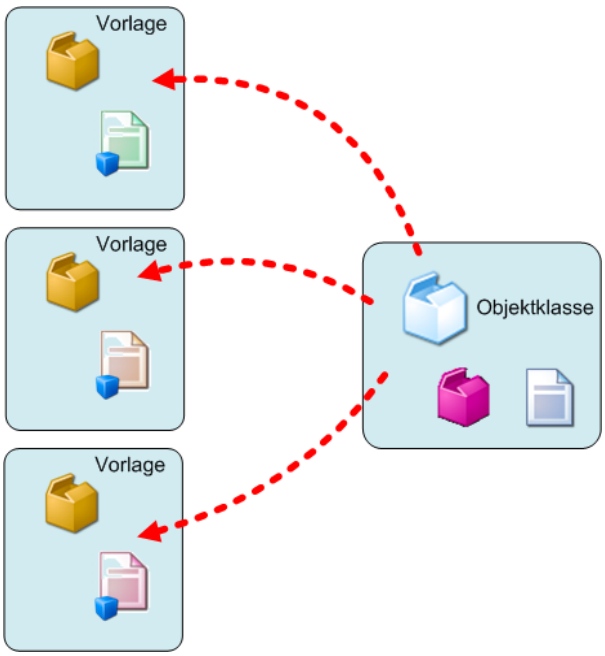
Hinweis

- Es empfiehlt sich Vorlagen-Objekte getrennt von den Objektklassen in der Struktur abzulegen.

Verwalten mehrerer Vorlagentypen in einer Objektklasse

In der Praxis wird es oft vorkommen, dass ein neues Dokument aus unterschiedlichen Vorlagedateien erstellt werden kann. Aus einer allgemeine Win-Word Objektklasse sollen z.B. unterschiedliche Dokumentarten (Brief, Protokoll,..) erzeugt werden. Dabei möchte man jeweils entsprechend aufgesetzte Vorlagen verwenden.

Für diesen Fall wird für jede Dokumentart ein separates Vorlagenobjekt erzeugt. Diese Dokument-Vorlageobjekte sind nichts anderes als Instanzen der eigentlichen Objektklasse mit zugeordneten Dokumentobjekten.



Im Unterschied zur Vorlagedatei der Objektklasse sind diese Vorlagen als normale Dokumentobjekte in *i-engine* vorhanden. Die Vorlagendateien werden auf einem speziellen Verzeichnis z.B. unter *Hauptklasse/_Mandant/Vorlagen* gespeichert.

Da die Klasse eines neuen Objektes durch die verwendete Vorlage gegeben ist, können mit dieser Methode auch neue Objekte unterschiedlicher Objektklassen erzeugt werden. Dazu stellen Sie auf dem Vorlagenverzeichnis einfach die unterschiedlichen Klassen als Vorlagen zur Verfügung.

Da alle neuen Objekte den gleichen Prozessablauf verwenden, eignet sich dieses Verfahren insbesondere für die einheitliche Erzeugung neuer Office-Dokumente (Excel, PowerPoint, WinWord etc.).

Hinweis

- Die Auswahl der gewünschten Vorlage erfolgt über einen entsprechenden BODY0 Formularteil im Attributprofil.

Versionsverwaltung

Um mehrere Varianten eines Dokumentes zu verwalten, bietet *i-engine* eine Versionskontrolle. Von einem bestehenden Dokument kann eine neue Version erzeugt und zusammen mit vorherigen Versionen verwaltet werden. Die Versionsnummer des neuen Dokumentes wird automatisch um Eins erhöht.

Im Versionsindex können die Versionsbezeichnungen vorgängig definiert werden. Damit können für den Versionsindex selbsterklärende Bezeichnungen wie "Testversion", "Marktversion" etc. festgelegt werden. Die Tabelle ist in der zugehörigen Objektklasse des Dokumentes abgelegt.

Die Definition der Versionsdaten erfolgt direkt in der Tabelle P_INDEX.

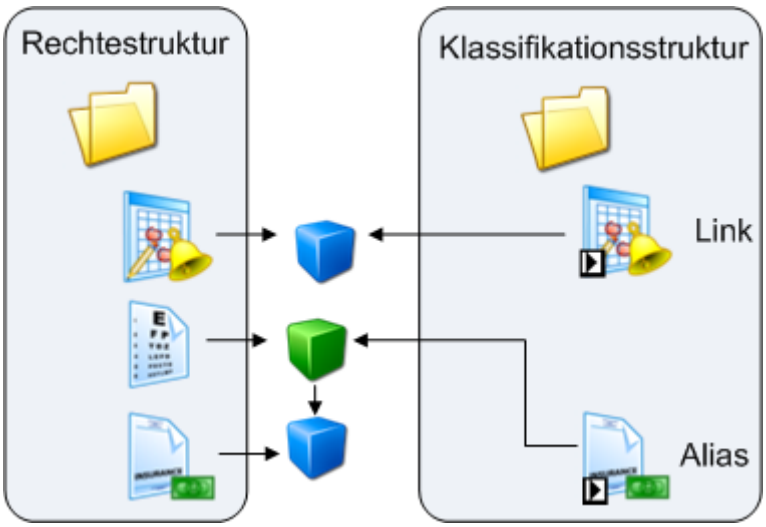
Feld	Eintrag
OBJCLASS_ID	ID der Objektklasse
INDEX_SEQ	Reihenfolge der Versionen
OBJ_INDEX	Bezeichnung der Version

Hinweis

- In den Eigenschaften der Objektklasse wird im Feld *Max Index* die gewünschte Anzahl Versionen festgelegt. Dieser Wert kann jederzeit abgeändert werden.
- Die Eigenschaften der neuen Version werden aus dem bestehenden Dokument übernommen – allerdings nur von jenen Feldern, die bereits in der Vorlage vorbesetzte Einträge hatten.

Wozu sind Link/Alias Objekte?

Damit Objekte von unterschiedlichen Benutzern gemeinsam verwendet werden können, kennt *i-engine* «Links» und «Alias» Objekte. Beide Elemente stellen eine Verknüpfung zwischen einem Eintrag und dem gemeinsamen Objekt dar und werden mit den Methoden *object.createAlias()* resp. *object.createNewlink()* erzeugt. «Links» werden in Listen kursiv dargestellt.



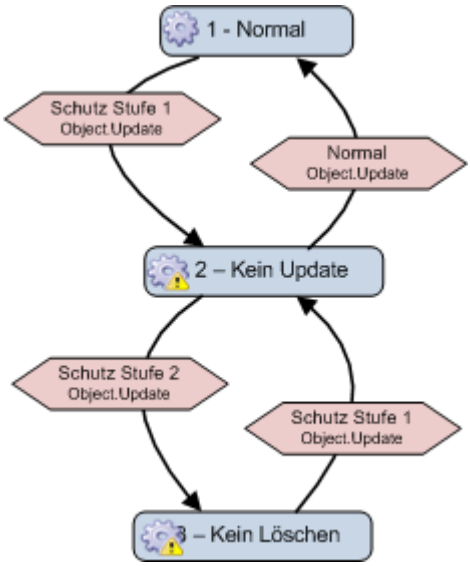
- «Alias» ist ein eigenständiges Objekt, welches auf das verknüpfte Objekt verweist. Das Objekt hat einen eigenen Prozess, der vom referenzierten Objekt unabhängig ist. Alle Objekt-Eigenschaften können individuell aufgesetzt werden. Damit kann das gleiche Objekt z.B. unter verschiedenen Bezeichnungen zur Verfügung gestellt werden.
- Bei «Links» handelt es sich nicht um ein eigenständiges Objekt, sondern lediglich um eine strukturelle Verknüpfung (eingetragen in der Tabelle STRUCTURE) zwischen zwei Objekten - Zielcontainer -> Originalobjekt. Der «Link» hat keine eigenen Rechte oder Eigenschaften.
- Die Versionierung von Objekten bietet umfangreiche Unterstützung auch für Links.
- Das Erzeugen eines Links oder Alias muss durch entsprechende Aktivitäten auf dem Originalobjekt und dem Zielcontainer implementiert sein. Links können auch direkt in der Tabelle STRUCTURE aufgesetzt werden:
  LOWER: Originalobjekt ID
  UPPER: Zielcontainer, wo der Link erscheinen soll
  TYPE: «LINK»

TREE: «YES» oder «NO», entsprechend dem Anzeigewunsch in der Baumanzeige.

- In einer Baumanzeige erscheint der «Link» Eintrag nur, wenn auch das Objekt angezeigt wird - dh. die notwendigen Rechte für die Anzeige des Objektes gegeben sind. Im Unterschied dazu sind «Alias» Einträge unabhängig.

Alias Objekte

Den «Alias» Objekten ist ein Prozess hinterlegt, mit welchem das Verhalten gesteuert werden kann.



Der Standardprozess unterstützt zwei Schutzstufen, mit welchen ein Verändern und/oder Löschen solcher Referenzeinträge verhindert werden kann. Damit kann vermieden werden, dass Referenzen, welche z.B. auf spezielle Versionen eines Objektes angelegt wurden, einfach nachgeführt werden.

Bemerkungen

- Im Kontextmenü werden Aktivitäten, welche sich auf das Alias-Objekte beziehen (z.B. Schutzstufe verändern) kursiv dargestellt - im Unterschied zu den normal angezeigten Aktivitäten, die sich auf das eigentliche Objekt beziehen.
- In der der Anzeige wird das Symbol der Statusanzeige zusammengesetzt:
  Symbol der sich aus dem Status des eigentlichen Objektes ergibt
  Symbol welches sich aus dem Status des Alias Prozess ergibt

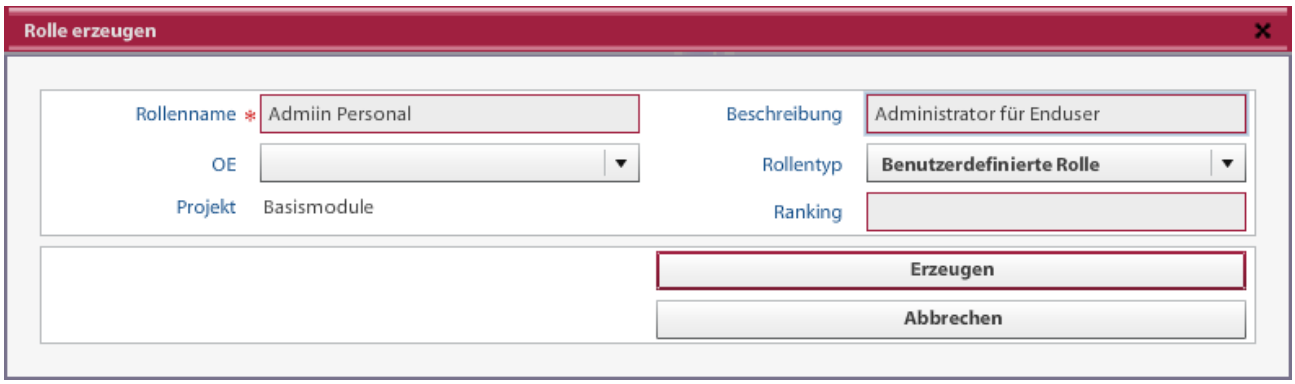
  Die Symbole werden in den Dateien: menu_schutz1, ... zur Verfügung gestellt.
- Im Unterschied zu «Links» können Änderungen an «Alias» Objekten protokolliert werden.

Rollen und Benutzer

Rollen erzeugen

Benutzerrolle

Zur generellen Strukturierung und Rechteverwaltung von Benutzern werden Benutzerrollen eingesetzt. Diese werden im Bereich «Basismodule / Rollen» abgelegt. Über das Kontextmenü **Rolle erzeugen** wird die Eingabemaske angezeigt.



Feld	Bezeichnung
Rollenname Beschreibung	Entsprechender Name und Beschreibung
Rollentyp	«Benutzerdefinierte Rolle»
Ranking	Wird nicht verwendet

Projektrollen

Zur Organisation von Rechten in Projekten und Aktenplänen/Dossiers werden Projektrollen verwendet. Diese sind oft auch unter «Basismodule / Rollen» abgelegt; oder im Projektbereich.

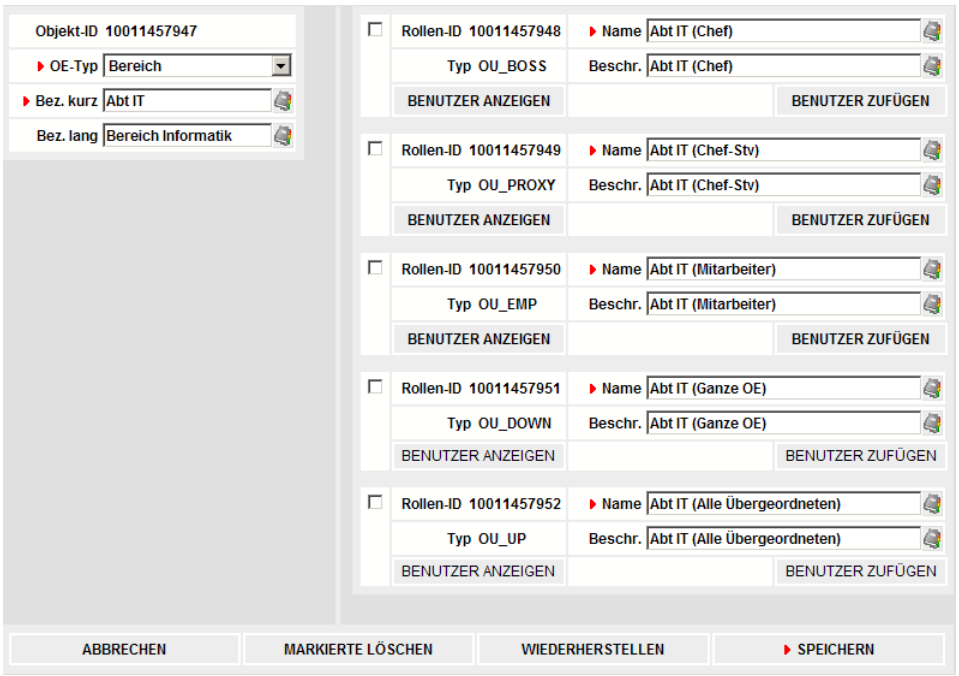
Feld	Bezeichnung
Rollenname	Entsprechender Name und Beschreibung

Feld	Bezeichnung
Beschreibung	
Rollentyp	«Projekt Rolle»
Ranking	Diese Zusatzinformation wird zur Dokumentation der Berechtigungsstufe verwendet. Die Angabe hat keine direkte Auswirkung auf *i-engine*. Der Wert kann aber im Customizing ausgewertet werden (z.B. Sortierung der Einträge). Besitzer Rolle: «70» Besitzer Stv Rolle: «50» Teammitglied: «30» Betrachter: «10»

Hierarchierollen

Mit den Hierarchierollen wird die OE-Struktur der Organisation abgebildet. Diese Rollen werden unter «Kundendaten / Stammdaten / Organisation» abgelegt.
Über das Kontextmenü **OE Erzeugen** wird die erste Eingabemaske aufgerufen.

Mit der Schaltfläche **Erzeugen** wird die OE mit den 5 Standardrollen erzeugt.



Feld	Bezeichnung
OE_Typ	Art der Organisationseinheit. Vorgesehen sind:

Feld	Bezeichnung
	Unternehmen; Unternehmensbereich, Geschäftseinheit, Fachbereich, Dienststellen, Bereich, Stab
Bez. kurz	Bezeichnung der Organisationseinheit
Bez. lang	Wird nicht verwendet
Rollentypen	Es werden folgende Hierarchierollen automatisch in die OE eingefügt: Chef: OU_BOSS Stellvertreter: OU_PROXY Mitarbeiter: OU_EMP Ganze OE: OU_DOWN Übergeordnete OE: OU_UP
Name, Beschr.	Für jede Rolle wird die Bezeichnung festgelegt.

Weitere Rollen

- Systemrollen: bspw. *Teilnehmer*, welcher bei einem Distribute die beteiligten Benutzer zugewiesen werden.
- Privatrolle
- Persönlich Rolle: wird bei Stellvertretung weitergegeben

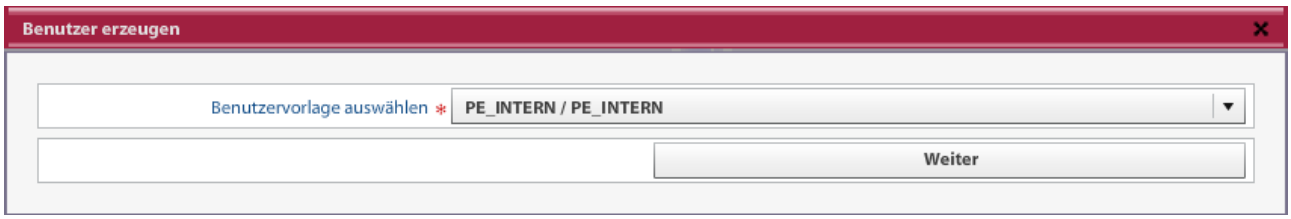
Benutzer erzeugen

Einen neuen Benutzer in *i-engine* einführen, bedeutet

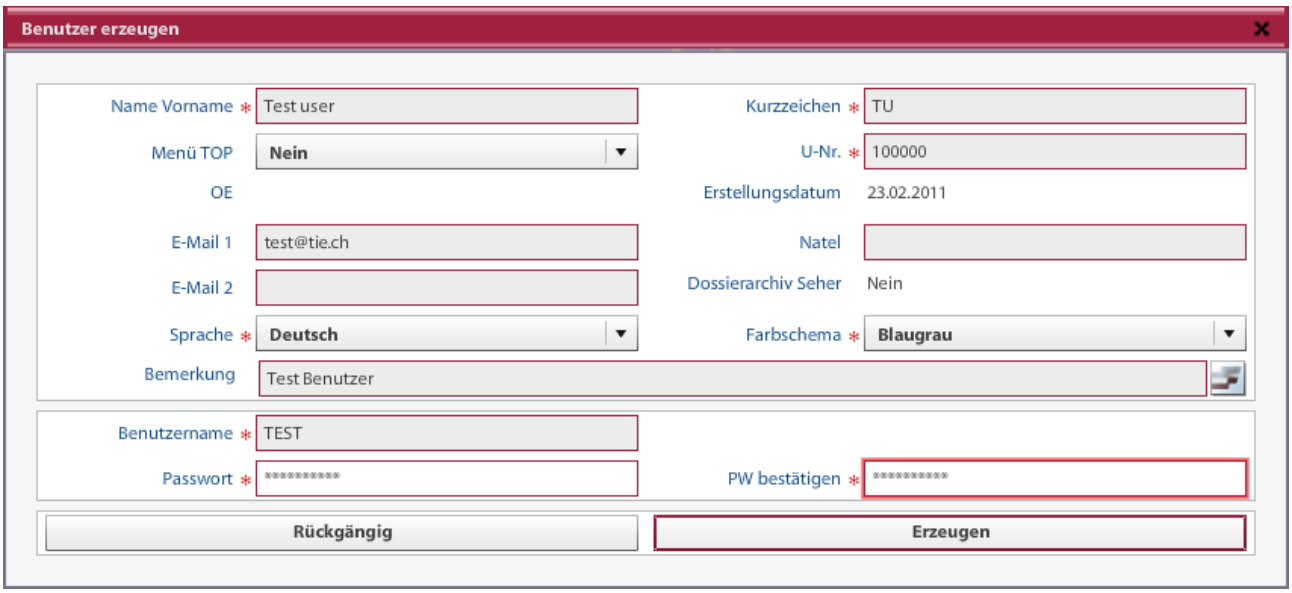
- Erzeugen des Benutzers – manuell oder per Import
- Zuweisen von weiteren Rollenmitgliedschaften

Erzeugen eines neuen Benutzers

Das Verfahren für das Erzeugen eines Benutzers wird anwendungsspezifisch Im Rahmen des Customizings implementiert. Die Benutzer sind im Ordner «Kunde / Stammdaten / Alle Benutzer» abgelegt. Das Anlegen eines neuen Benutzers wird über das Kontextmenü **Neuer Benutzer** ausgelöst. Hier ein Implementierungsbeispiel:



Für die Erzeugung eines neuen Eintrages wird oft eine Vorlage verwendet, bei welcher Eigenschaften wie Rollenmitgliedschaften oder die persönliche Ordnerstruktur bereits aufgesetzt sind. In einem ersten Schritt wird die gewünschte Vorlage ausgewählt.

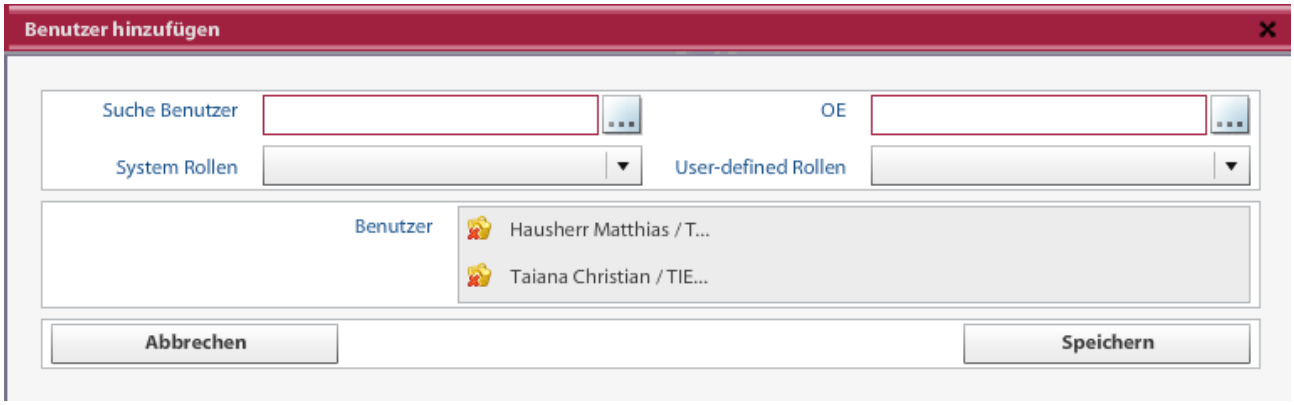


Im Folgeschritt werden die Informationen zum Benutzer erfasst. Die Daten sind anwendungsabhängig.

Für eine automatisierte Übernahme von Benutzerdaten ist eine detaillierte Spezifikation der Schnittstelle notwendig. Je nach Lösung (Verbundene Tabelle/Import-Script) wird der Vorgang durch das Customizing-Team oder die TIE-Entwicklung realisiert.

Weitere Informationen im Abschnitt «*Datenimport* (Seite 216)».

Der Rolle einen Benutzer zuweisen



Prinzipiell kann einer Rolle ein bestimmter Benutzer oder aber eine andere Rolle zugeordnet werden. In diesem Fall ist zu beachten, dass nur eine 1-stufige «Rolle-in-Rolle» Zuweisung unterstützt wird. Der Aufbau der Eingabemaske für den Kunden ist Teil des Customizing.

Hinweis

- Der Benutzer hat immer seine Persönliche Rolle und seine Privat-Rolle zugewiesen.
- Ein Benutzer sollte nicht mehr als 1000 Rollenmitgliedschaften aufweisen. Sonst sind längere Wartezeiten beim Anmelden zu erwarten. Wenn Projektrollen für die Verwaltung von Dossierberechtigungen eingesetzt werden, kann dies für Vorgesetzen-Positionen leicht erreicht werden. In diesem Fall sollte die Berechtigung für den Vorgesetzten eher mit speziellen Benutzerrollen gelöst werden.
- Im Feld USER_TYPE der Tabelle P_USER wird der Benutzertyp aufgesetzt.
  «USER»: Normaler Benutzer
  «IMPORTER»: Bei Importvorgängen ist das Nachführen der Vorgangsliste oder das Generieren von E-Mails unerwünscht. Dieser Benutzertyp ist deshalb für Batch-Benutzer zu verwenden.

Die Benutzervorlage

Üblicherweise wird für das Erfassen neuer Benutzer eine kundenspezifische Benutzervorlage erstellt. Dazu wird ein «normaler» Benutzereintrag unter «Kunde > Stammdaten > Alle Benutzer > Virtuelle Benutzervorlagen» erstellt.

- Der Benutzer wird als virtueller Benutzer erzeugt, damit er nicht irrtümlich auch für das Anmelden eingesetzt werden kann.
- ggf. ist die Benutzererfassungsmaske resp. die Aktivität **Benutzer erzeugen** auf dem Ordner anzupassen, so dass weitere Eigenschaften erfassbar sind.
- Für die Benutzervorlage wird die Menüleiste aufgesetzt. s. Abschnitt «*Menüleiste & Strukturanzeige* (Seite 149)»
- Die Strukturanzeige des persönlich Ordners ist durch *i-engine* aufgesetzt. In speziellen Situationen kann aber der Strukturbaum durch direkte Veränderungen in der Tabelle STRUCTURE entsprechend modifiziert werden. z.B. direkten Zugriff auf einen Ordner einfügen.

Passwort Einstellungen

Das Password-Handling wird durch Oracle vorgenommen. Die Einstellungen für die Berechtigungen sind deshalb mit dem Oracle Management Werkzeug oder über entsprechende SQL-Scripts vorzunehmen.

Im Register *Kennwort* können Sie die Parameter für das Account-Kennwort angeben. Dieses Register enthält folgende Elemente:

Kennwortablauf

ABLAUF IN	Begrenzt die Anzahl von Tagen, nach denen ein Kennwort abläuft. Wählen Sie einen Wert aus der Dropdown-Liste, oder geben Sie einen bestimmten Wert an.
SPERRE	Begrenzt die Anzahl von Tagen, während denen ein Kennwort nach der ersten erfolgreichen Anmeldung nach dem Ablauf des Kennworts geändert werden kann.

Kennworthistorie beibehalten

BEIBEHALTEN	Gibt an, wie oft ein Kennwort geändert werden muss, bevor es erneut benutzt werden kann. BEIBEHALTEN WÄHREND ist deaktiviert, wenn in diesem Feld ein Wert angegeben wird.
BEIBEHALTEN WÄHREND	Begrenzt die Anzahl von Tagen, bevor ein Kennwort nach dem Ablauf wieder verwendet werden kann. BEIBEHALTEN ist deaktiviert, wenn in diesem Feld ein Wert angegeben wird.

Kennwortkomplexität durchsetzen

Ermöglicht die Verwendung einer PL/SQL-Routine für die Bestätigung des Kennworts, wenn sich Benutzer, denen dieses Profil zugewiesen wurde, bei einer Datenbank anmelden. Diese PL/SQL-Routine muss lokal zur Ausführung auf der Datenbank verfügbar sein, für die dieses Profil gilt. Oracle stellt ein Standardskript (utlpwdmg.sql) bereit. Sie können jedoch auch eine eigene Routine erstellen oder eine Software von einem anderen Hersteller als Alternative verwenden. Die Routine zur Überprüfung des Kennworts muss SYS gehören.

NULL	(keine Kennwortbestätigung) ist standardmässig eingestellt.

Account bei nicht erfolgreicher Anmeldung sperren

SPERREN NACH	Begrenzt die Anzahl von nicht erfolgreichen Anmeldeversuchen, bevor ein Benutzer aus dem Account gesperrt wird.
SPERREN WÄHREND	Gibt an, wie viele Tage das Account gesperrt wird, nachdem die angegebene Anzahl von Anmeldeversuchen nicht erfolgreich verlaufen ist. Wenn UNBEGRENZT angegeben wird, kann nur der Datenbankadministrator die Sperre des Accounts aufheben.

Hinweis

- Wenn STANDARD für eine der Kennwortoptionen ausgewählt ist, werden die in dem Profil STANDARD definierten Werte benutzt.
- Wird der Datenchecker «checkPasswords» eingesetzt, ist zu beachten, dass die Anforderungen an das Passwort abgeglichen werden.

Fenster

Hauptfenster

Initialer Bildaufbau

Die Aufteilung der Anzeige beim Aufrufen der *i-engine* Anwendung ist vorgegeben und kann nicht angepasst werden. Was in der linken Strukturanzeige oder in der rechten Datenanzeige erscheint, ist durch den ersten Menüeintrag in der Menüleiste definiert.

Die Menüleiste

Die Menüleiste kann für jeden Benutzer individuell aufgesetzt werden. s. dazu Abschnitt «*Menüleiste und Strukturanzeige* (Seite 149)». Häufig wird der persönliche Ordner als erstes angezeigt. Hinweise über dessen Einstellungen sind auch im Abschnitt «Die Benutzervorlage» zu finden.

Anpassen von Farben, Schriften oder Logo

Farbaspekte oder Schriftarten und –grössen sind in einer CSS-Datei festgelegt. Änderungen werden in der paradocs.css Datei vorgenommen. Details s. Spezialdokumentation.

Das Firmenlogo wird in der Datei «logo.gif» bereitgestellt. Die Grafik sollte folgende Bedingungen erfüllen:

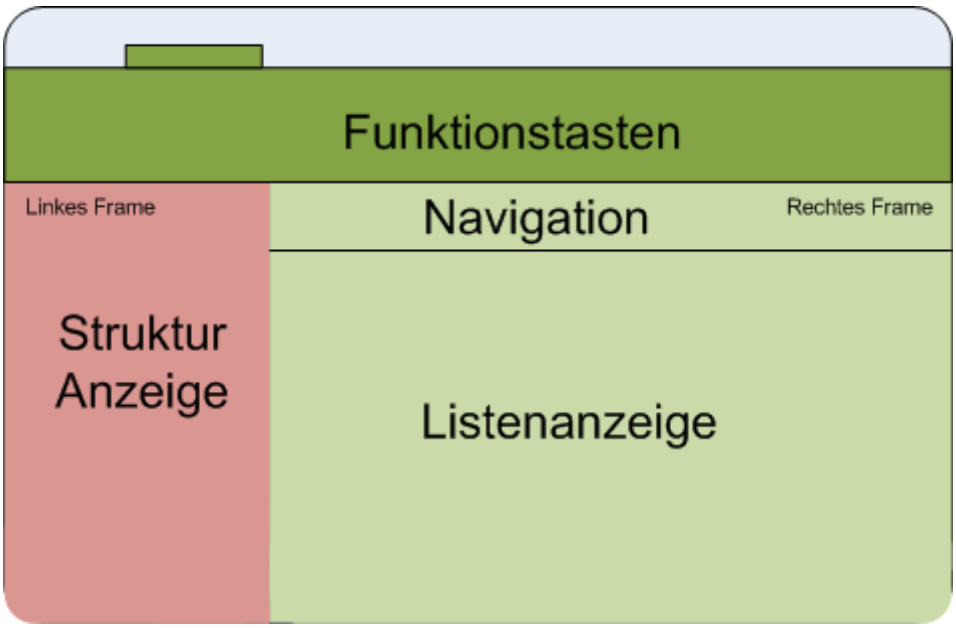
- Höhe exakt 60 Pixel, Breite beliebig
- Transparenter Hintergrund

Projektfenster

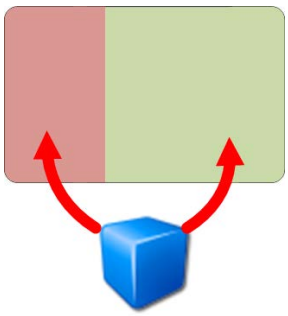
DB Action

Projektfenster anzeigen

Das Projektfenster kann nur über die Methode *Project.Edit()* angezeigt werden. Das Projektfenster wird als neues Hauptregister im angezeigten Fenster eröffnet. Der Aufbau des Fensters ist fix. Im Attributprofil kann nur der Teil «Informationen» definiert werden. Die verfügbaren Funktionstasten werden in der jeweiligen Aktivität festgelegt. Dazu werden die beiden Felder *Tasten Position vertikal* resp. *horizontal* aufgesetzt.



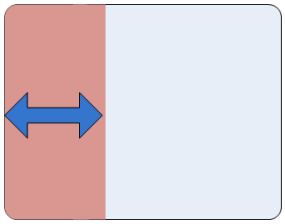
Die Aktivitäten der Funktionstasten beziehen sich immer auf das ursprüngliche Projekt-Objekt, auf welches die Methode *Project.Edit()* angewandt wurde. Sollen andere Objekte angesprochen werden, muss dies über die Methode *Object.Execute()* implementiert werden.



Anzeigemethode beim Laden

Die Anzeigemethode, welche beim Laden des Objektes in die beiden Frames verwendet werden muss, kann abgeändert werden. Die Einstellungen werden in der Tabelle OBJECT_PROPERTY vorgenommen.

Tabellenfeld	Eintrag
OBJ_ID	Aktivitäts-ID, welche das Projektfenster öffnet. z.B. *Object.ProjectEdit()*
KEY / VALUE	«CONTENT_START_METHOD_ID» Methoden-ID die beim Laden in das rechte Frame zum Einsatz kommen soll. Es wird die erste Aktivität verwendet, in welche die Methode eingesetzt wird. Fehlt dieser Eintrag, so wird die Methode *Object.ObjList()* verwendet.
KEY / VALUE	«CONTENT_START_REFERENCE» Sollte die Methode in mehreren Aktivitäten des aktuellen Prozesszustandes verfügbar sein, muss die betreffende Aktivität über einen Referenzstring selektiert werden. Der Referenzstring ist auch in der Aktivität einzutragen.
KEY / VALUE	«TREE_START_METHOD_ID» Methoden-ID die beim Laden in das linke Frame zum Einsatz kommen soll. Es wird die erste Aktivität verwendet, in welche die Methode eingesetzt wird. Fehlt dieser Eintrag, so wird die Methode *Object.Tree.ObjList()* verwendet.
KEY / VALUE	«TREE_START_REFERENCE» Auswahl der Aktivität falls die Methode in mehreren Aktivitäten des aktuellen Prozesszustandes verfügbar ist.
SELECT	Über eine Dropdown-Liste können andere Objekte im gleichen Container direkt ausgewählt werden. «YES»: Dropdown Auswahlliste anzeigen «NO»: nicht anzeigen



Spaltenbreite linkes Frame

Im Projektfenster kann die gewünschte Spaltenbreite für das linke Anzeigeframe festgelegt werden. Die Einstellungen werden in der Tabelle OBJECT_PROPERTY vorgenommen.

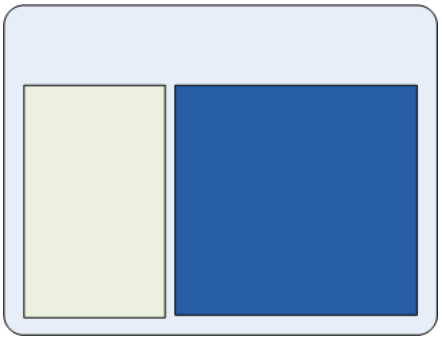
Tabellenfeld	Eintrag
OBJ_ID	Aktivitäts-ID, welche das Projektfenster öffnet. z.B. *Object.ProjectEdit()*

Tabellenfeld	Eintrag
KEY / VALUE	«TREE_WIDTH» Breite des linken Frames entweder prozentual «xx%» oder absolut in Pixel «pp».

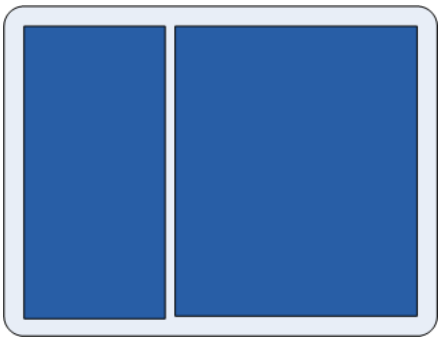
Festerframes

Die Struktur – Fensterframes – wird durch die eingesetzte Methode bestimmt. Hier ein Überblick über die Möglichkeiten in *i-engine*.

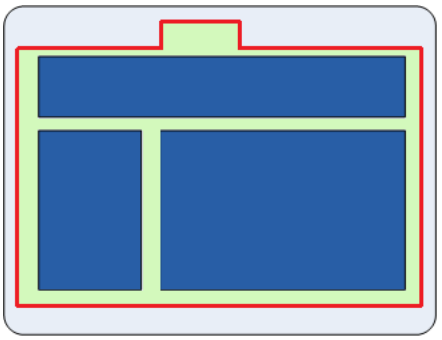
Object.ObjList()
Constraint.ObjList()
Object.ConstraintObjList()



Object.2Frames()



Project.Edit()



Eingabeelemente

Im Attributprofil können neben reinen Textfeldern auch erweiterte Eingabeelemente eingesetzt werden. Alle Felder sind in der Tabelle ATTRIBUTE_SHOW definiert. Über zusätzliche Tabellenfelder werden die erweiterten Eingabeelemente aufgesetzt.

	Sprachabhängige Felder TAG = «TEXT_ML»
	Dropdown-Liste TAG = «SELECT» POPUP_OBJ_ID = Objekt-ID des hinterlegten SQL-Statements Das SQL-Statement (vom SQL-Typ «POPUP») liefert die Liste der Einträge für Dropdown Anzeige. Die Return-Tabelle des SQL-Statements ist aufgebaut nach <Speicherwert>|<Anzeigewert>
	Kalender TAG = «DATE»
	Optionstaste TAG = «RADIO» POPUP_OBJ_ID = Objekt-ID des hinterlegten SQL-Statements Das SQL-Statement (vom SQL-Typ «POPUP») liefert die Liste der Optionen. Die Return-Tabelle ist aufgebaut nach <Speicherwert>|<Anzeigewert>. Für jedes Datenpaar wird ein Optionsknopf angezeigt.
	Sucher TAG = «FINDER» Das Vorgehen beim Erstellen eines Suchers ist im Abschnitt «*Sucher definieren* (Seite 277)» beschrieben.
	Pop-up TAG = «POPUP» Das Vorgehen beim Erstellen eines Popup-Fensters ist im Abschnitt «*Pop-up Fenster definieren* (Seite 288)» beschrieben.

Anzeige einer statischen Auswahl

Für Optionstasten und Dropdown-Listen werden oft statische Werte eingesetzt. Zur Anzeige von statischen Werten haben sich zwei Verfahren bewährt:

Für einfachste Listen ist eine direkte Erzeugung via DUAL Tabelle geeignet:

```
SELECT 'Y','Ja' FROM Dual
UNION ALL
SELECT 'N','Nein' FROM Dual
```

Grössere Auswahllisten oder mit längeren Texten werden über Wertetabellen oder Einträge in die Tabelle P_CODE realisiert.

Einträge aus Wertetabelle

Kunden- oder anwendungstypische Listen werden über «Wertelisten» implementiert. Diese bieten den Vorteil, dass sie vom Anwender ohne direkte Datenbank-Manipulation modifiziert und erweitert werden können. Details s. «*Abschnitt Werteliste* (Seite 58)» erzeugen.

Einträge in der Tabelle P_CODE

Statische, vom Basissystem vorgegebene Listen werden durch Wert aus der P_CODE Tabelle erzeugt. Jede Zeile eines Eingabeelementes bedingt einen Eintrag in dieser Tabelle. Der Wert wird durch den SQL-Befehl entsprechend aufgesetzt:

```
SELECT Internat_Value,Display_Value_Short FROM Sys_P_Code
WHERE Tab_Name='Test Popup'
```

TAB_NAME	Gemeinsamer Name für die Menüzeilen des betreffenden Eingabeelementes. Muss eindeutig gewählt werden.
SEQ	Reihenfolge der Einträge in Pop-up Menü
INTERNAL_VALUE	Wert, welcher bei der Auswahl des Menüpunktes verwendet werden soll.
DISPLAY_VALUE	Text des Menüeintrages.
SHORT & _LONG	Es besteht die Möglichkeit zwei unterschiedliche Längen zu definieren. Welcher konkret verwendet wird, hängt natürlich vom SQL-Befehl ab.
_L1/_L2/_L3	Es können max. 4 Sprachvarianten festgelegt werden.
DESCR	wird nicht verwendet

Tipp

In der P_CODE Tabelle sind bereits die meisten, oft benötigten Texte vorhanden. Deshalb sollten Sie zuerst prüfen, ob Sie wirklich einen neuen Text haben.

Textanpassungen

i-engine unterstützt die Verwendung von 4 Sprachen. Alle Textstrings können 4-sprachig erfasst und abgespeichert werden. Jeder Benutzer kann unter **Einstellungen** seine bevorzugte Sprache auswählen.

Es gilt standardmässige folgende Zuordnung:

Normales Feld	Deutsch
L1_ Feld	Französisch
L2_ Feld	Italienisch
L3_ Feld	Englisch

Integration

Single Sign-On

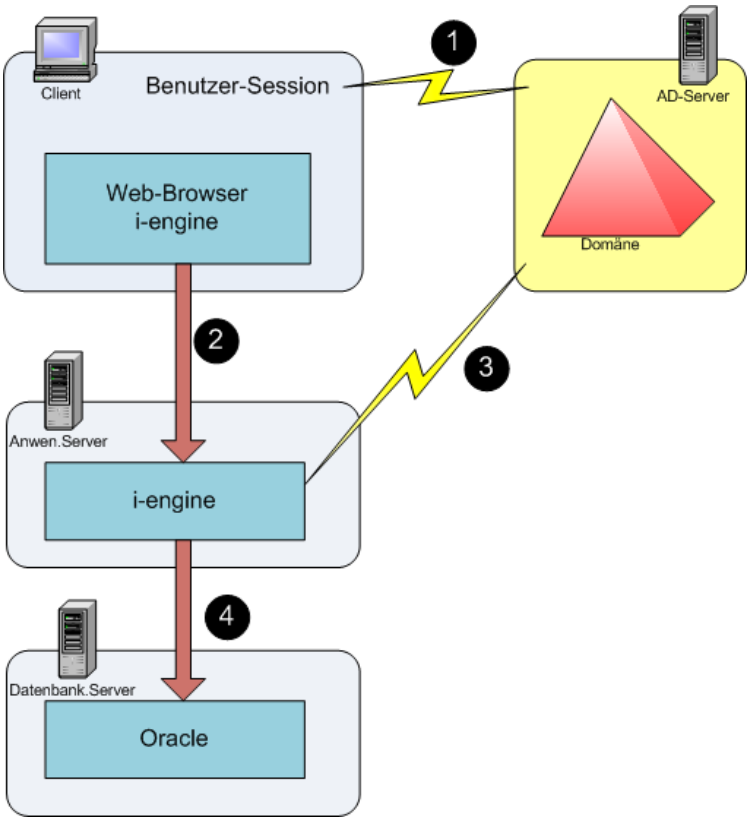
Im Anmeldedialog kann statt der Eingabe von Benutzerdaten das automatische Login durch Single Sign-On verwendet werden. Zudem kann über das spezielle Anmeldescript «login-spnego.jsp» die Anwendung auch ohne Anmeldemaske aufgerufen werden.

Wie funktioniert SSO?

Vereinfacht dargestellt werden beim Single-Sign-On (SSO) folgende Schritte durchgeführt:

1. Der Benutzer meldet sich an seinem System an. Dabei wird die Anmeldeinformation durch das Domain AD-System überprüft. Im Erfolgsfall erhält die aktive Session ein Berechtigungs-Ticket vom AD-System zurück.

2. Der Benutzer ruft nun im Web-Browser die *i-engine* Anwendung auf. Dabei wird das aktuelle Berechtigungsticket an *i-engine* übermittelt.



3. Diese lässt das erhalten Berechtigungsticket durch das AD-System validieren.

4. Bei gültigem Ticket erfolgt der Zugriff auf die Oracle-Datenbank durch einen allgemeinen Proxy-Benutzer.

Hinweise

- SSO kann bspw. basierend auf dem älteren NTLM-Verfahren (ausschliesslich Windows) oder dem aktuellen Kerberos-Verfahren durchgeführt werden.Das Verfahren ist unabhängig von *i-engine* und wird durch den Applikationsserver vorgegeben.
- Die Unterstützung von SSO durch *i-engine* bedingt, dass der Applikationsserver entsprechend aufgesetzt und konfiguriert wird..

Zusatzdaten in Benutzerdaten

DB Action

Für den Einsatz von SSO müssen die einzelnen Benutzereinträge in der Tabelle P_USER ergänzt werden.

Feld	Bedeutung
EXT_AUTH_NAME1	Benutzername in AD-Directory Wird beim älteren NTLM-Verfahren verwendet.

Feld	Bedeutung
	Aufbau: domain\username
EXT_AUTH_NAME2	Benutzername in AD-Directory Wird beim aktuellen Kerberos-Verfahren verwendet. Aufbau: name@domain

Üblicherweise ist bestimmt, welches Authorisierungsverfahren in einer Umgebung eingesetzt wird. In heterogenen Umgebungen kann aber auch das NTLM-Verfahren als Fall-Back bei nicht erfolgreichen Kerberos Versuchen definiert werden.

Spezielle Konfiguration Client-seitig

Die Voraussetzung für ein funktionsfähiges SSO-Verfahren ist, dass der Web-Browser das Berechtigungsticket an die *i-engine* übermittelt. Dies macht der Internet Explorer nur wenn er entsprechend instruiert wird.

1. Die *i-engine* Site muss in die Liste der Vertrauenswürdigen Websites aufgenommen werden.

2. In den Sicherheitseinstellung des IE muss definieren werden, dass er sich bei vertrauenswürdigen Websites automatisch anmelden soll.

Hinweis: Mit Firefox ist kein SSO basierend auf NTLM oder Kerberos möglich.

Authentifikation mit iTAN

iTAN steht für **i**ndizierte **T**r**an**saktionsnummer und ist eines von vielen Verfahren, die eine Transaktionsnummer für die Authentifikation am System als zusätzliche Sicherheitskomponente implementiert. Hierbei wird dem Benutzer ein Brief zugestellt, der ein Koordinatensystem mit TANs beinhaltet (iTAN Kärtchen). Nach dem Login mit Benutzernamen und Passwort verlangt das System zusätzlich eine TAN, wobei das System definiert, welcher der TANs im Koordinatensystem tatsächlich eingegeben werden muss.

i-engine - Authentifizierung
Meldung
Bitte geben Sie die iTAN Nummer ein (Gross- und Kleinschreibung sind nicht von Bedeutung):
F1 (Liste vom 14.12.2009):
LOGIN
tag $customer$ ($timestamp$)

Beispiel des iTAN Abfrage Dialogfensters, welches nach dem üblichen Login-Fenster angezeigt wird.

Der Login Prozess

- Nach drei fehlgeschlagenen Anmeldeversuchen wird das Benutzerkonto gesperrt und es muss vom Administrator manuell entsperrt werden.
- Der iTAN Prüfalgorithmus ignoriert Gross- und Kleinschreibung sowie alle Leerzeichen.
- Sind mehrere iTAN Authentifikationen für einen Benutzer aktiv, so wird immer die älteste aller Listen verwendet.

Um Flexibilität im Einsatz von iTAN zu gewährleisten wurden verschiedene *login* JSP-Prozeduren definiert. Neben dem bekannten *login.jsp* gibt es neu das *login_itan.jsp*, welches zusätzlich zur normalen Benutzernamen / Passwort Abfrage anschliessend die iTAN abfragt.

Wenn die Anmeldung mit iTAN erzwungen werden soll, müssen JSPs, welche eine Anmeldung ohne iTAN ermöglichen, mittels Proxy / Firewall entsprechend abgesichert werden. Die Zugriffkontrolle auf die verschiedenen JSPs ist eine kritische Komponente um den Zugriff auf die *i-engine* mittels iTAN zu sichern.

docbox Integration

Für die Kommunikation mit der docbox und Partnern, welche die Authentifizierung des Benutzers bereits vollzogen haben wurde die JSP-Prozedur *invoke_itan_authenticated.jsp* erstellt. Hierbei wird vom Benutzer nur noch die iTAN

verlangt, vorausgesetzt, das aufrufende System liefert den Benutzernamen als Parameter mit.

iTAN Kärtchen erstellen

Die Codematrix wird dem Benutzer üblicherweise per Post zugestellt. Für die Produktion wird die Methode Object.XML2Ext verwendet. Für die Produktion der Kärtchen im pdf-Format steht eine spezielle Vorlagen (10080734) bereit. Dabei werden die generierten iTAN XML-Daten mittels xsdl-Transformation in eine pdf-Datei konvertiert.

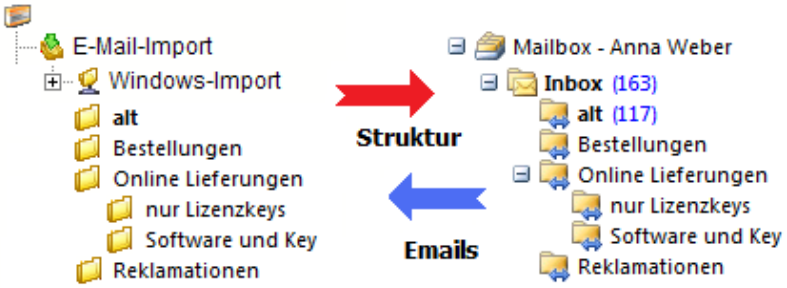
iTan Liste vom **19.08.2010**

	A	B	C	D	E	F	G	H	J	K
1	NHQH32	RCU8JF	CC2EP4	BQ2WBE	NEFTNU	MA4UAC	FUJTJU	W2CV4F	VF9ND3	7MBPYU
2	K32R3D	ARZTBM	7B3FDM	ELZY2U	K2778Y	BQ83YY	973JJX	QYJWCE	QY89R2	AKM479
3	LTP9ZB	3JBH7D	38QRKH	NZKUA2	PQXE9P	BEJCCT	KHXKDP	WKE7J2	2WKYK8	HQMK3L
4	YLKNX9	JV7KBJ	MLRRCB	4EAERY	P79239	DR83PM	D34NKC	ZU2QYD	WXBRWE	BDKPNK
5	BUEPD8	2UQRTR	X8T3PM	TUR3LN	FAWLF9	UEKY8R	R3P9E3	7JQADF	CXMYNK	3LD7W9
6	2TEEWJ	BNVYNV	YL37FL	LCDUAM	DB8HLF	2BT48W	HAUFJE	9KMUE7	97CHFU	M3TPLH
7	92R4PW	8PFQ4M	8QMJ9P	CCY3M4	U2VKFY	D84BX2	4VM3FB	J2XARJ	VZRTP9	E4DQNN
8	CDFET9	X3PYUD	ZW7YRL	3HTVFD	T4EQM2	9ZF249	HDNJUU	BB3FYL	7CA3LW	9WF8AC
9	FXCZM7	LFBFKN	LAKPMK	Y3YNNL	EN27MN	PKKU8X	TZYY4H	YAULP3	JQUM93	9VRUHW
10	WXTMMA	FBD7C3	UWF482	VPTWMV	4U8KHQ	8TLT47	NLXYNU	2FACRW	F3HNCN	NMP27W

Exchange/Outlook Anbindung

Über das IMAP-Protokoll können E-Mails und Termine automatisch in *i-engine* importiert werden. Die Daten werden in einem festgelegten Intervall vom Exchange-Server eingelesen und abgelegt. In den Ablage-Objekten sind die notwendigen Prozess hinterlegt, welche ggf. die Weiterverarbeitung automatisch anstossen oder den Benutzer orientieren.

in *i-engine* wird im «IMAP» Objekt die benötigte Struktur für die Ablage der E-Mails festgelegt – z.B. Dossiermappen. In Outlook/Exchange wird diese Struktur automatisch repliziert. In Outlook eingehende E-Mails können nun durch den Benutzer oder Regeln auf die Verzeichnisse verteilt werden.



Einrichten IMAP

DB Action

Das E-Mail-Import Objekt (Objektklassentyp «IMAP») wird für jeden Benutzer durch die Vorlage aufgesetzt. Die Einstellungen werden in der Tabelle IMAP vorgenommen.

Feld	Bedeutung
IMAP_ID	Objekt ID des E-Mail-Import Objektes (vorgegeben)
USER_NAME	Die Einstellungen sind individuell für jeden Benutzer
IMAP_HOST	Exchange Host Adresse – z.B. tie.ch
IMAP_USER	Exchange Anmeldung – Outlook User
IMAP_PSWD	Exchange Anmeldung – Outlook Passwort
INTERVAL	Import Zeitintervall in Sekunden Generell werden beim Anmelden an *i-engine* die E-Mails abgeholt. «99000000»: manuelle Abfrage
IMAP_PORT	Standard sind 143 resp. 993 für SSL
USE_SSL	«0»: SSL nicht verwenden «1»: SSL verwenden

Hinweis

- Das manuelle Einlesen der E-Mails wird über das Stellvertreter-Symbol ausgelöst:

WebDav Anbindung

Über das WebDAV-Protokoll können Dateien automatisch in *i-engine* übernommen werden. Die Daten werden im gewählten Zeitintervall vom definierten Server eingelesen und abgelegt. In den Ablage-Objekten sind die notwendigen Prozess hinterlegt, welche ggf. die Weiterverarbeitung automatisch anstossen oder den Benutzer orientieren.

In *i-engine* wird im «WEBDAV» Objekt die benötigte Struktur für die Ablage der Dateien festgelegt – z.B. Dossiermappen. Im WebDAV Server wird diese Struktur automatisch repliziert. Auf dem Server können die Dateien durch den Benutzer oder Scripts auf die Verzeichnisse verteilt werden.

Einrichten WebDav

DB Action

Das WebDAV-Import Objekt (Objektklassentyp «WEBDAV») wird für jeden Benutzer durch die Vorlage aufgesetzt. Die Einstellungen werden in der Tabelle WEBDAV vorgenommen.

Feld	Bedeutung
WEBDAV-ID	Objekt-ID des WebDAV-Import Objektes.
USER_NAME	Die Einstellungen sind individuell für jeden Benutzer.
WEBDAV_USER	User für WebDav Server.
WEBDAV_PSWD	Passwort für WebDav Server.
INTERVAL	Import Zeitintervall in Sekunden. Generell werden beim Anmelden bei *i-engine* die Daten abgeholt. «99000000»: manuelle Abfrage

Zusätzlich wird der WebDAV-Server in der Tabelle PROPERTIES aufgesetzt.

Feld	Bedeutung
USER_NAME	Fix «SYSTEM»
KEY / VALUE	Key: «WEBDAV_URI» Adresse des WebDAV-Servers. z.B. «http://172.16.15.236/webdav/»
KEY / VALUE	Key: «WEBDAV_PATH» Pfadangabe zum Hauptordner der Importordnerstruktur z.B. «D:\data\»

Zugriff von der Arbeitsstation

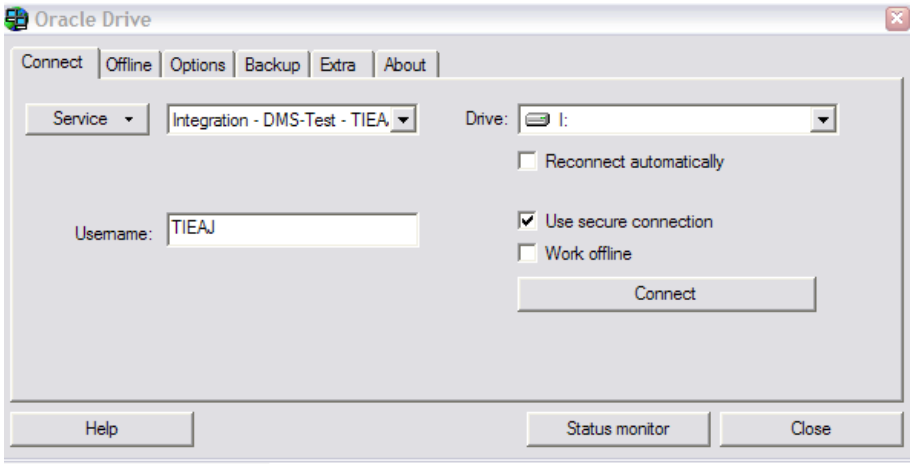
Oft ist auch ein Zugriff von der Arbeitsstation auf die vom WebDAV Server vorgehaltenen Daten notwendig. Für diesen Zugriff kann z.B. OracleDrive oder Microsoft WebFolder eingesetzt werden. Die Software muss entsprechend konfiguriert werden.

Konfigurieren OracleDrive

Bitte beachten Sie, dass Oracle-Drive eine lizenzpflichtige Applikation der Firma Oracle ist. Nach der Installation von OracleDrive sind folgende Konfigurations-schritte notwendig:

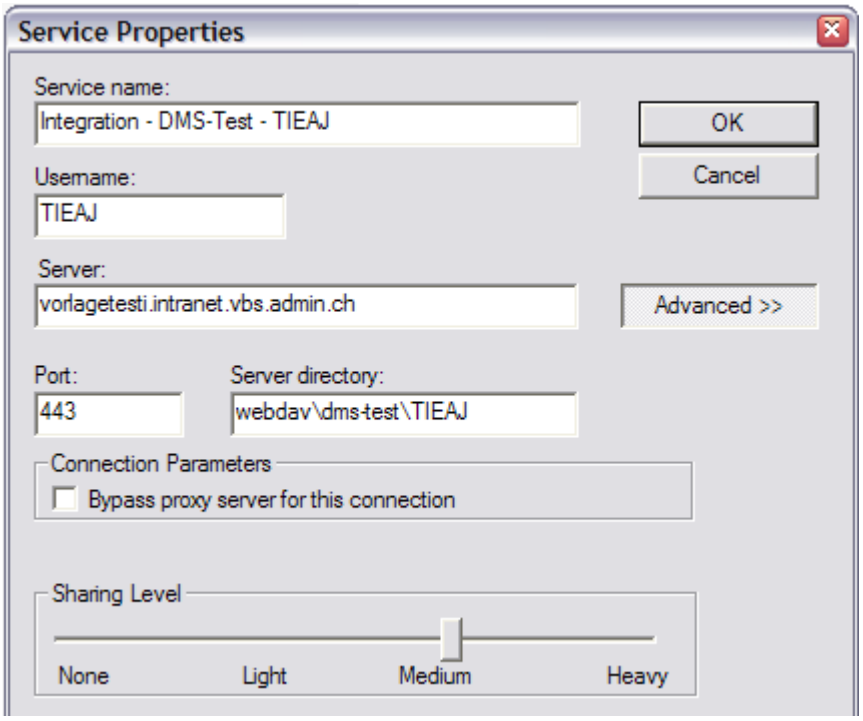
1. Wählen Sie über die Schaltfläche **Service** die Funktion «New» aus.

2. Schalten Sie «Use secure connection» ein.



3. Erweitern Sie die Maskenanzeige über die Schaltfläche **Advanced**. Geben Sie die notwendigen Daten ein.

4. Übernehmen Sie die Einstellungen mit **OK**.

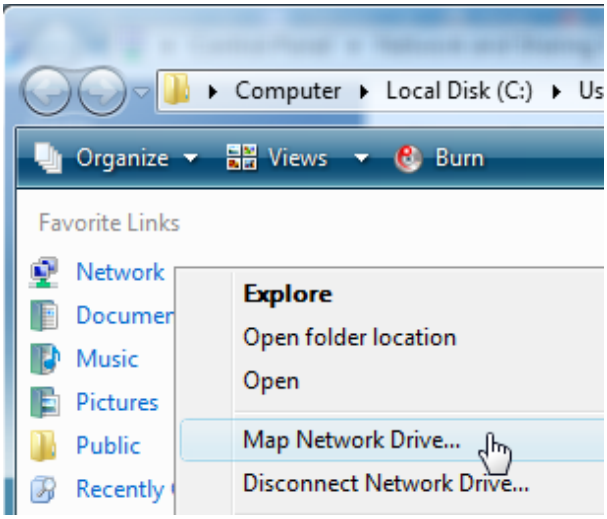


Konfigurieren Windows WebFolder

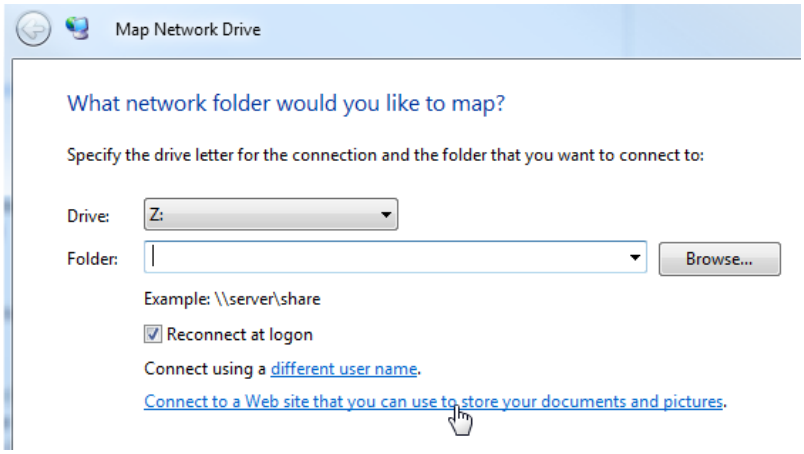
Alle modernen Varianten von Microsoft Windows werden mit der Unterstützung von WebFoldern ausgeliefert.

Die Definition einer Verbindung auf die Daten des WebDAV-Servers ist vom Betriebssystem abhängig. Unter Windows 98 und XP konnte man sehr einfach über den Internet Explorer WebDAV Ordner «verbinden» und dann lokal nutzen. Unter Windows Vista sieht die korrekte Einrichtung von WebDAV Laufwerken wie folgt aus:

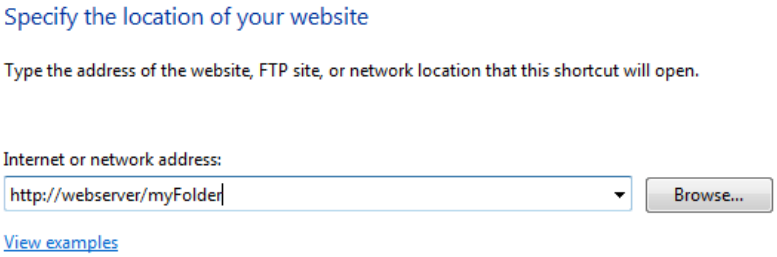
Rechtsklick auf «Netzwerk» im Explorer und im Kontextmenü «Netzwerklaufwerk zuordnen…» auswählen



Fahren Sie fort mit der Funktion «Verbindung mit einer Website herstellen, auf der Sie Dokumente und Bilder speichern können».


Map Network Drive
What network folder would you like to map?
Specify the drive letter for the connection and the folder that you want to connect to:
Drive:
Z:
Folder:
Browse...
Example: \\server\share
Reconnect at logon
Connect using a different user name.
Connect to a Web site that you can use to store your documents and pictures.

Ein Assistent führt Sie nun weiter durch das Definieren eines WebFolders. Geben Sie nun die Adresse des WebDAV-Servers ein. z.B. z.B. «http://172.16.15.236/webdav/»


Specify the location of your website
Type the address of the website, FTP site, or network location that this shortcut will open.
Internet or network address:
http://webserver/myFolder
Browse...
View examples

Evtl. wird anschliessend die Eingabe von Benutzername und Passwort wie gewohnt erforderlich. Damit hat man unter Windows Vista wie gewohnt einen WebDAV Ordner verbunden.

Datenimport

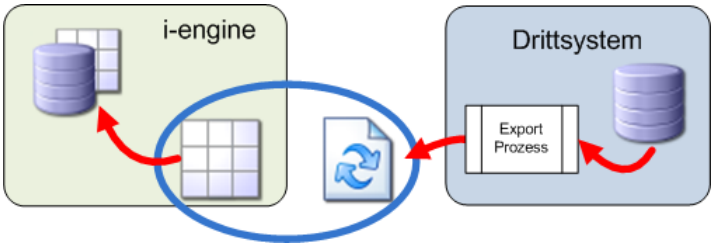
i-engine bietet die Möglichkeit Daten von Drittsystemen wie ERP oder PIM einmalig oder regelmässig zu übernehmen. Generell wird der Datenaustausch über eine Datei vollzogen, welche das Drittsystem aufbereitet und der *i-engine* zur Verfügung stellt. Zur Verarbeitung der bereitgestellten Daten bieten sich die beiden folgenden Verfahren an:

- Verknüpfung der Importdatei in die Oracle Datenbank.
- Importprozess zum Einlesen der Datei und Abspeichern der Werte.

Welche Methode zum Einsatz kommt, hängt z.B. davon ab

- Vorverarbeitung der Daten notwendig.
- Realisierung durch Customizing- oder TIE-Entwicklerteam.
- Anforderungen an Übernahmeintervall, Datenmenge etc.

Verknüpfte Tabellen



Dieses Verfahren eignet sich für regelmässige Datenübernahmen ab einem Drittsystem. Dazu werden im Drittsystem die Daten aufbereitet und in einer csv-Datei bereitgestellt. Die Datei kann jeweils alle Daten oder nur geänderte und neue Daten enthalten.

Oracle bietet die Möglichkeit externe csv-Dateien als verknüpfte Tabellen einzubinden. Da diese Tabellen jedoch nur sequentiell durchsucht werden können, wird in einem ersten Schritt der Inhalt der verknüpften Tabelle in eine richtige Oracle-Tabelle umkopiert.

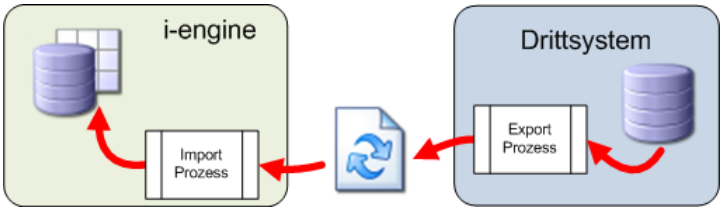
Im nächsten Schritt übernimmt eine PL/SQL-Prozedur die Abarbeitung der angelieferten Daten.

Hinweis

- Ein Löschen ist in *i-engine* nicht vorgesehen. Handelt es sich z.B. bei den Informationen um Benutzer, so werden die nicht mehr gemeldeten Benutzer inaktiv gesetzt.

- Die Bereitstellung der Daten und das Abarbeiten müssen auf einander abgestimmt werden. Die Datenübernahme muss als Aufgabe eingeplant werden. s. dazu Abschnitt «*Zeitplanen von Aufgaben* (Seite 237)».

Importprozedur



Das Drittsystem produziert eine Datendatei (csv, xml,..), deren Daten von einem Importprozess in *i-engine* gelesen und abgelegt werden. Im Zuge des Systemdesigns werden die notwendigen Spezifikation und Übernahmeintervalle spezifiziert. Die Implementierung kann nur vom TIE-Entwicklerteam vorgenommen werden.

Hinweis

- Die Bereitstellung der Daten und der Importprozess müssen auf einander abgestimmt werden. Der Importprozess muss als Aufgabe eingeplant werden. s. dazu Abschnitt «*Zeitplanen von Aufgaben* (Seite 237)».

HL7 Schnittstelle

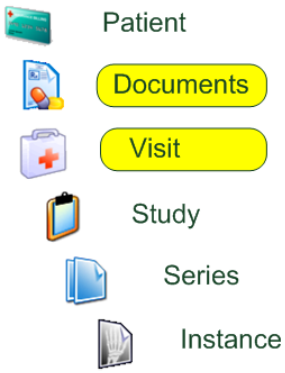
Health Level 7 (HL7) ist ein internationaler Standard für den Austausch von Daten zwischen Computersystemen im Gesundheitswesen. HL7 bietet Interoperabilität zwischen Krankenhausinformationssystemen (KIS), Praxisverwaltungssystemen (PVS), Laborinformationssystemen (LIMS), Systemen zur Leistungsabrechnung und dem *i-engine* Dokumentmanagementsystem. Die Implementierung des HL7-Importers ist kundentypisch und kann nicht parametrisiert werden.

Für die Datenübernahme im HL7-Format kann das Drittsystem eine Datei übermitteln oder die Daten in eine spezielle Tabelle in Oracle schreiben. Der Importprozess wird TIE-seitig individuell auf die angebotenen und benötigten Daten angepasst. Auch der Export von Daten aus *i-engine* ist vorgesehen.

Über die HL7-Schnittstelle werden üblicherweise

- Patienten- und Falldaten - von einem KIS-System
- Patienten-Dokumente - von Umsystemen wie Labor, Kardiographie,..

übermittelt. Die Patienten- und Falldaten (Visit) werden durch den HL7-Importer in die Standard Patientenstruktur übernommen. Dokumente werden direkt unter dem Patienten abgelegt.



Eintreffende Falldaten enthalten auch den jeweils zugeordneten Patienten. Der HL7-Importer erstellt die Visit-Einträge unter dem Patienten. Sollte der Patient noch nicht existieren, wird automatisch ein neuer eröffnet.

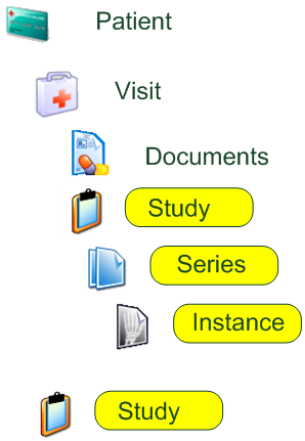
Unabhängig vom HL7-Datenaustausch treffen die Study/Series/Instance-Daten über die DICOM-Schnittstelle ein. Diese enthalten eine Zuordnung zum Fall. Ist der betreffende Fall noch nicht vorhanden, werden die Informationen vorerst unter «Studienordner» abgelegt. Der HL7-Importer prüft deshalb auch, ob er dem neuen Visit-Eintrag ggf. Study-Daten aus dem Verzeichnis «Studienordner» zuweisen kann.

Ungültige HL7-Daten - ungültiger Patienten-ID - werden in einem speziellen Ordner «Unzuweisbar» (10011471191) abgelegt.

DICOM Schnittstelle

«Digital Imaging and Communications in Medicine» (DICOM) ist ein offener Standard zum Austausch von Informationen in der Medizin. DICOM standardisiert sowohl das Format zur Speicherung der Daten, als auch das Kommunikationsprotokoll zu deren Austausch. DICOM ist auch die Grundlage für die digitale Bildarchivierung in Praxen und Krankenhäusern. Die DICOM Schnittstelle kann in beide Richtungen benutzt werden.

Die Patientenstruktur von *i-engine* entspricht weitgehend der entsprechenden DICOM Struktur. Allerdings ist die Struktur mit einer zusätzlichen Stufe für Fälle (Visit) erweitert. Die Fall-ID wird in praktisch allen Implementationen als zusätzliche PRIVATE Feld übermittelt. Über die DICOM-Schnittstelle werden primär die Daten zu Study, Series und Instance angeliefert. Aber auch Informationen zum Patienten können geliefert werden. Diese werden jedoch durch HL7-Daten jederzeit überschrieben, da das KIS-System der eigentliche Ursprung der Daten ist.



Der DICOM-Importer übernimmt die vom Drittsystem angelieferten Datensätze. Jeder Datensatz enthält auch die zugewiesenen übergeordneten Objekte. Bspw. ein «Series» Objekt enthält den Verweis auf die übergeordnete «Study» und den «Patienten». Der DICOM-Importer ordnet das neue Objekt automatisch entsprechend den übergeordneten Referenzen in die Patientenstruktur ein. Falls ein übergeordnetes Element noch fehlt, wird es automatisch erstellt. Ein fehlender Patienten-Eintrag wird durch die Anlieferung entsprechender Visit-Daten erzeugt. Study-Informationen, die noch nicht einem Fall zugeordnet werden können, werden vorerst in einem separaten Study Ordner abgelegt.

Die Patient- und Visit-Objekte werden typischerweise über die HL7-Schnittstelle angeliefert. Dies kann erst später geschehen, so dass ein referenzierte Fall noch nicht vorhanden ist. Die betreffende Study kann damit noch nicht unter dem Fall abgelegt werden, sondern wird in einem Verzeichnis «Studienordner» abgelegt, das sich ebenfalls unter dem Patienten befindet.. Trifft später der betreffende Fall ein, wird der Eintrag automatisch zum fall verschoben.

Bei fehlerhaften Datensätzen - kein Patienten-ID, kein Fall-ID - sind die Informationen nur in der Tabelle HL7MESSAGE abgelegt und können dort ggf. ausgewertet werden.

Konfigurierung der DICOM Schnittstelle

DB Action

Die Anpassung auf die kundenspezifische Datenübermittlung wird in einem Attributprofil (10090033) vorgenommen. Das Attributprofil enthält einige Standardeinträge, die nicht angepasst werden müssen. Die Einträge des *Show-Typs* METADATA definieren den Recordaufbau:

Feld	Beschreibung
TAG_TYPE	«METADATA»

DISPLAY_NAME	DICOM Code Für «PRIVAT» Felder bleibt dieses Feld leer.
ATTRIBUTE_NAME	internes Feld
DATATYPE	Wird für die korrekte Sortierung der Textstrings verwendet. «TEXT» «NUMBER» «DATE»
DESCR	Feldbeschreibung
TAG	«PUBLIC»: Feld entspricht dem DICOM Standard «PRIVATE» Zusätzliches Feld, bspw. FID

Integration in HTML-Seite

Direkter URL Aufruf

Ein *i-engine* Objekt kann direkt über einen URL Aufruf angezeigt werden. Diese Möglichkeit wird oft in E-Mail-Meldungen verwendet. Der Empfänger hat so über einen einfachen Klick direkten Zugriff auf das entsprechende Dokument. Für die Anzeige wird die HTML-Ansicht verwendet.

Prinzipieller Aufbau

```
http://<server><port>/invoke.jsp?login_page=logininvoke.jsp
<parameter>
```

<server>	Adresse z.B. «www.spital.ch»
<port>	Portnummer, z.B. «:80»
invoke.jsp	Genereller Einstiegspunkt für die Behandlung der Anfrage.
<parameter>	In der Parameteraufzählung werden die notwendigen Aufrufparameter übergeben Einige Aufrufparameter sind immer notwendig: parent_id: «0»
	Anmeldung
login_page=logininvoke.jsp	Mit der Verwendung von «logininvoke.jsp» wird die Abfrage der Anmeldeinformation vermieden. Die notwendigen Angaben können über die <parameter> mitgegeben werden. Mit der Routine «login_itan.jsp» ist eine Anmeldung nach dem iTAN Verfahren möglich. Die Benutzerangaben können auf zwei Arten übermittelt werden:
US, PW	Benutzername und Passwort im Klartext angeben. **Aus Sicherheitsgründen ist diese Variante nur in Spezialfällen einsetzbar.**
urlkey	Ein Systemprozess (J_URLKEY) kreiert in einem festgelegten Zeitintervall - üblicherweise alle 2 Min - einen Sicherheitscode. Dieser ist in der Tabelle PROPERTIES unter dem Key=«URLKEY» abgelegt und wird dort von der Drittanwendung

	abgefragt. Der Wert wird im URL-Aufruf mitgegeben. Diese Verfahren kann nicht für E-Mails verwendet werden, da zum Zeitpunkt der Ansicht der Schlüssel abgelaufen sein wird.
	Objektauswahl Die Auswahl des Objektes kann direkt spezifiziert oder über einen Sucher ausgewählt werden.
	Objekt spezifizieren
obj_id	Objekt auf welchem die angegeben Aktivität oder Methode ausgeführt werden soll.
activity_id oder invoke_methode_id invoke_Reference	
next_methode_id	Idealerweise wird dem Benutzer das Dokument automatisch angezeigt. Im aktuellen Prozessstatus des Objektes muss dazu eine entsprechende Aktivität zur Verfügung stehen – mit Berechtigung für den betreffenden Benutzer. Durch «next_methode=xxxxx» kann die ID der in der Aktivität verwendeten Methode angegeben werden.
menu	Anzeige des Kontextmenüs «YES»/«NO» Bspw. bei der Anzeige einer Objektliste kann es notwendig werden, auch das Kontextmenü mit den Aktivitäten anzuzeigen.
	Sucher spezifizieren
finder_id	Gewünschten Sucher Liefert die Ergebnisliste nur ein einziges Objekt wird dieses anstelle der Liste angezeigt. In diesem Fall sind auch die weiteren Parametereinstellungen verfügbar.
next_target	Ein in der Liste angeklicktes Dokument wird oft in

einem separaten Fenster angezeigt. Durch Festlegen des Zielfensters wird dies erreicht.

«next_target=_blank»

Weitere Parameter

<feld> — Alle in der Objektklasse verfügbaren Felder können als Parameter übermittelt werden.

z.B. temp.t1; sys_attribute.t1

Beispiel:

```
http://www.spital.ch/invoke.jsp?login_page=logininvoke.jsp
      &urlkey=SLUKT5YCYWSXNKATW2C&US=TEST&fildee_id=1009002
1
      &sys_attribute.t1=90021255&temp.all=YES&temp.vip=YES
      &invoke_methode_id=10010151&invoke_reference=URLAURUF
_K1
```

Hinweis

- Im Rahmen von kundenspezifischen Lösungen ist auch das direkte Anmelden eines externen Benutzers über den URL Aufruf möglich. Allerdings sind vorgängig die Sicherheitsanforderungen abzuklären.

KIS-Link - Integration in Klinik Informationssystem

Ein *i-engine* Objekt kann direkt über einen URL Aufruf angezeigt werden. Als KIS-Link wird eine Variante des URL-Aufrufs bezeichnet, welche bzgl. Leistung und Darstellung für die Integration in KIS Anwendungen speziell geeignet ist. So wird bspw. die Java-Engine erst geladen, wenn sie gebraucht wird - was zu schnelleren Ladezeiten führt.

Prinzipieller Aufbau

```
http://<server><port>/invoke.jsp?login_page=logininvoke.jsp
<parameter>
```

<server> — Adresse z.B. «www.spital.ch»

<port> — Portnummer, z.B. «:80»

invoke.jsp — Genereller Einstiegspunkt für die Behandlung der Anfrage.

<parameter> — In der Parameteraufzählung werden die notwendigen Aufrufparameter übergeben

Einige Aufrufparameter sind immer notwendig:
parent_id: «0»

Anmeldung

login_page=logininvoke.jsp	Mit der Verwendung von «logininvoke.jsp» wird die Abfrage der Anmeldeinformation vermieden. Die notwendigen Angaben können über die <parameter> mitgegeben werden. Mit der Routine «login_itan.jsp» ist eine Anmeldung nach dem iTAN Verfahren möglich. Die Benutzerangaben können auf zwei Arten übermittelt werden:
US, PW	Benutzername und Passwort im Klartext angeben. **Aus Sicherheitsgründen ist diese Variante nur in Spezialfällen einsetzbar.**
urlkey	Ein Systemprozess (J_URLKEY) kreiert in einem festgelegten Zeitintervall - üblicherweise alle 2 Min - einen Sicherheitscode. Dieser ist in der Tabelle PROPERTIES unter dem Key=«URLKEY» abgelegt und wird dort von der Drittanwendung abgefragt. Der Wert wird im URL-Aufruf mitgegeben. Diese Verfahren kann nicht für E-Mails verwendet werden, da zum Zeitpunkt der Ansicht der Schlüssel abgelaufen sein wird.

Objektauswahl

Die Auswahl des Objektes kann direkt spezifiziert oder über einen Sucher ausgewählt werden.

Objekt spezifizieren

obj_id	Objekt auf welchem die angegeben Aktivität oder Methode ausgeführt werden soll.
activity_id oder invoke_methode_id invoke_Reference	
next_methode_id	Idealerweise wird dem Benutzer das Dokument automatisch angezeigt. Im aktuellen Prozessstatus des Objektes muss dazu eine entsprechende Aktivität zur Verfügung stehen – mit Berechtigung für den betreffenden

	Benutzer. Durch «next_methode=xxxxx» kann die ID der in der Aktivität verwendeten Methode angegeben werden.
menu	Anzeige des Kontextmenüs «YES»/«NO» Bspw. bei der Anzeige einer Objektliste kann es notwendig werden, auch das Kontextmenü mit den Aktivitäten anzuzeigen.
	Sucher spezifizieren
finder_id	Gewünschten Sucher Liefert die Ergebnisliste nur ein einziges Objekt wird dieses anstelle der Liste angezeigt. In diesem Fall sind auch die weiteren Parametereinstellungen verfügbar.
next_target	Ein in der Liste angeklicktes Dokument wird oft in einem separaten Fenster angezeigt. Durch Festlegen des Zielfensters wird dies erreicht. «next_target=_blank»
	Weitere Parameter
<feld>	Alle in der Objektklasse verfügbaren Felder können als Parameter übermittelt werden. z.B. temp.t1; sys_attribute.t1

Vergleich

Hier die beiden Integrationsvarianten im Vergleich:

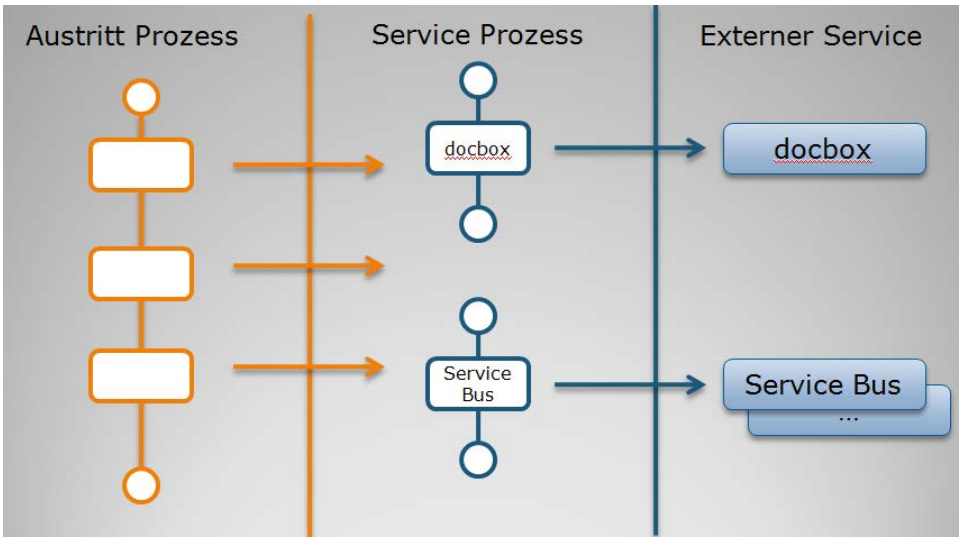
Eigenschaft	URL	KIS
Leistung	normal	schnelle Ladezeiten
Datenobjekte	flexibel	nur Objektlisten
Listendarstellung	normal	Spalten nur für definierte Aktionsymbole. Kein Status- und Objektklassensymbole

WebServices

i-engine kann Information mit Drittanwendungen austauschen. Abhängig von der Drittanwendung werden entsprechende Kommunikationsdienste wie Webservice (SOAP/REST) oder Java Message Service eingesetzt. Aktuell wird die Kommunikation mit **docbox** unterstützt.

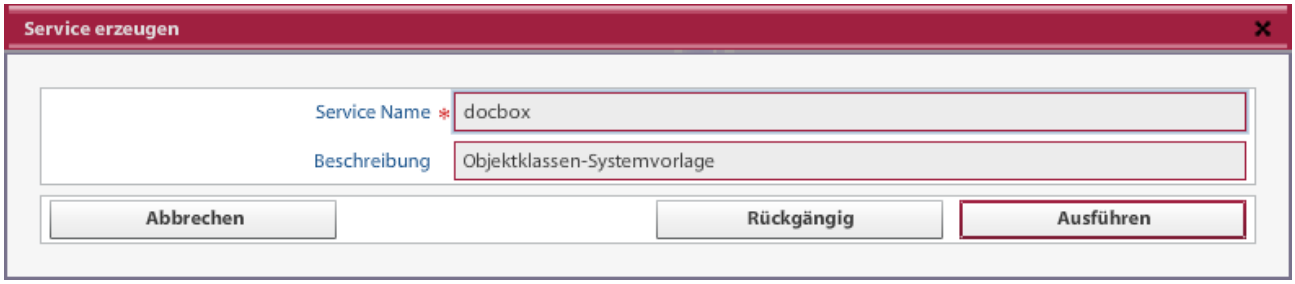
Zusätzlich zur vollständigen Definition des Service-Objektes müssen auch die XML-Message angelegt werden. Die zugehörigen XML-Dateien enthalten die Kommunikationsdetails für den Austausch eines bestimmten Dokumentes/Meldung mit dem Provider/Dienst. Der Inhalt der XML-Datei muss in einem vorgängigen Schritt anwendungsspezifisch aufgesetzt werden.

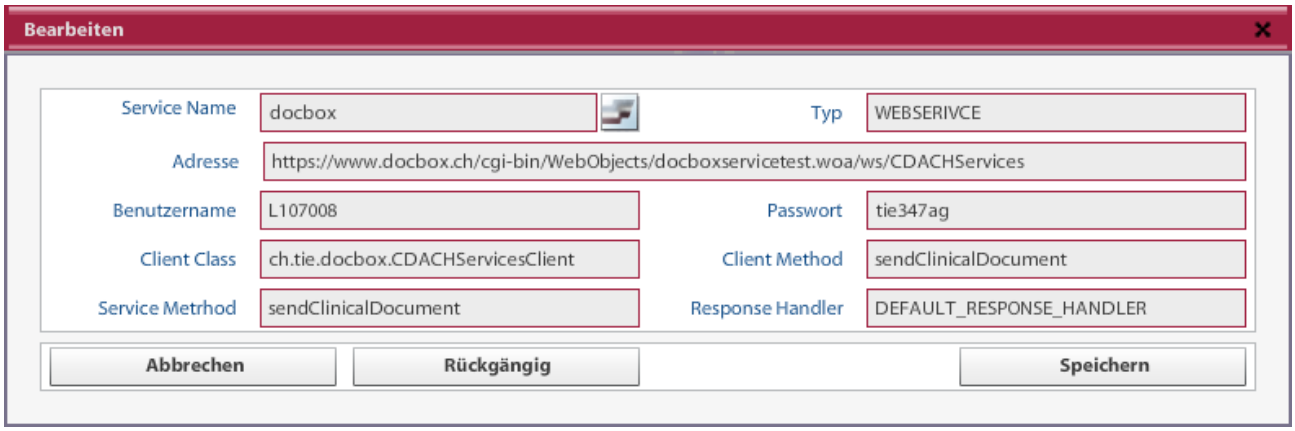
Wie im Detail die Daten ausgetauscht werden - z.B. nur Textmeldung, gezippte Datei,.. - ist durch die Implementierung in der Java-Klasse und/oder die Definitionen in der Message XML-Datei festgelegt.



Docbox Service erzeugen

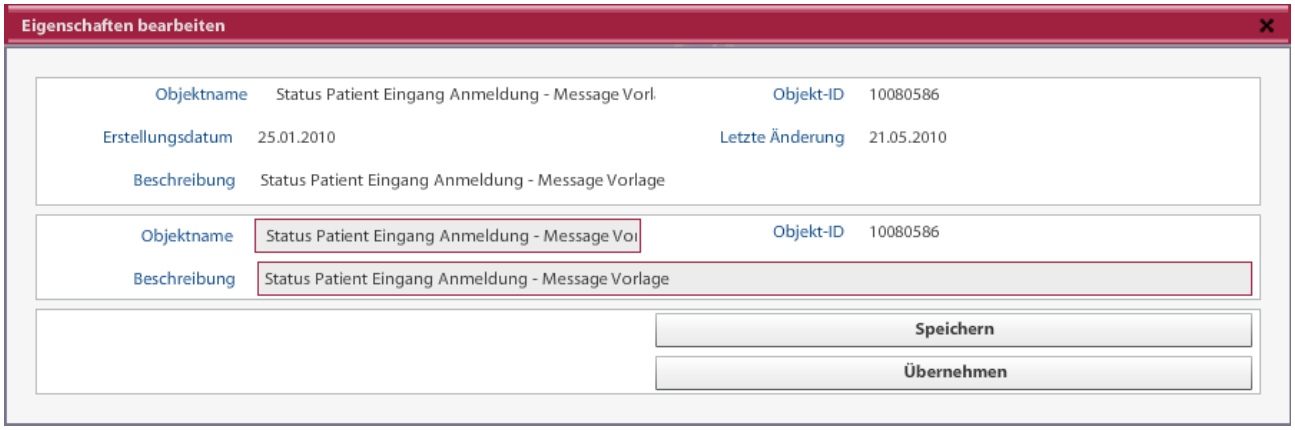
Zuerst muss der docbox Service erzeugt und konfiguriert werden.





Messages definieren

Zu jedem für den Austausch vorgesehenen Datenelement muss ein «WS Message» Objekt erzeugt werden.



Zu jedem Message-Objekt wird eine XML-Datei bereitgestellt, welche die eigentliche Definition des Datenaustausches auf Basis XML-File festlegt. Diese XML-Datei enthält die komplett applikationsabhängige Definition für jedes auszutauschende Datenfeld.

Datenaustausch initiieren

Der Datenaustausch kann vom Drittsytem über einen SOAP-Befehl oder von *i-engine* mittels der Methode *Service.Call()* ausgeführt werden.

Forms Handling

InfoPath

DB Action

Bei InfoPath handelt es sich um ein Windows-Anwendungsprogramm der Firma Microsoft, das sowohl die Gestaltung von XML-basierten Formularen, wie auch deren späteres Befüllen mit Informationen erlaubt. Die erstellten Formulare können über *i-engine* verteilt werden, so dass Anwender-Informationen einfach und benutzerfreundlich gesammelt und gespeichert werden können. InfoPath wird in Verbindung mit *i-engine* genutzt, um Anwendern Formulare für typische Situationen wie Status-Reports, Reisekostenabrechnung, etc. zur Verfügung zu stellen.

In der Tabelle ATTRIBUTE_SHOW wird die Beziehung zwischen den Formularfeldern und den *i-engine* Attributfeldern festgelegt.

Zuordnung InfoPath Feld zu Attributfeld

Feld	Bedeutung
SHOW_TYPE	«XML»
DISPLAY_NAME	InfoPath XML-Tag für Feld
ATTRIBUTE_NAME	Zugehöriges Attributfeld

Zuordnung Formularauswahl

Feld	Bedeutung
SHOW_TYPE	«XML»
DISPLAY_NAME	«TIE_ViewSteuerung»
ATTRIBUTE_NAME	Name des Attributes, welches die Bezeichnung des zu verwendenden InfoPath Formulars enthält.

Zuordnung Aktivitäten

Feld	Bedeutung
SHOW_TYPE	«XML»
DISPLAY_NAME	«TIE_ProzessSteuerung»
ATTRIBUTE_NAME	Name des Attributes, in welches der Rückgabewert der InfoPath Schaltfläche abgelegt wird.

Hinweis

Die Auswertung der erfassten Werte wird über entsprechende SQL-Statements implementiert, welche die Ergebnisse als ObjList ausgeben.

Spezielle Konfigurationsaufgaben

AutoUpdate

i-engine ist eine browserbasierte Web-Applikation. Um für *i-engine* auf dem Client Funktionalitäten bereitzustellen, die die Möglichkeiten des Webbrowsers überschreiten, werden signierte Java-Applets eingesetzt. Der Benutzer muss die Ausführung des Applets erlauben, damit *i-engine* Aktionen auf dem Client ausführen darf.

Diese Aktionen sind im Einzelnen:

- Anlegen von Installations- und Arbeitsverzeichnis im Home-Verzeichnis des Benutzers)
- Kopieren/ Löschen von Dateien (Dokumente) in das/ aus dem Arbeitsverzeichnis
- Interaktion mit dem Betriebssystem z.B. Ermitteln betriebssystemspezifischer Ordner
- Kopieren/Löschen betriebssystemspezifischer Hilfsanwendungen via Autoupdate
- Kopieren/Löschen kundenspezifischer Hilfsanwendungen via AutoUpdate

AutoUpdate beim Anmelden

Die Funktion «Autoupdate» kopiert Dateien auf den Client, welche zum Betrieb von *i-engine* auf dem Client-System benötigt werden. Die Dateien werden innerhalb von *i-engine* definiert und das Verhalten über Attribute (Zielverzeichnis auf dem Client, Art der Aktualisierung, Betriebssystem, Browser) gesteuert. Dateien können rollenspezifisch verteilt werden. Die Autoupdate Aktivitäten werden auf dem Client protokolliert und sind somit vollständig transparent für den Benutzer.

Die Funktionalität des Web-Browsers wird über ein spezielles Applet (MSmonitor) erweitert. Dieses wird nebst anderen Komponenten nach erfolgreichem Authentifizieren während des Einloggens auf dem Client installiert.

MSmonitor übernimmt:

- Das Durchführen der Dateiübermittlung beim Aus- und Einchecken der MS Office Dokumente.

- Das Starten der MS Office-Anwendung zur Bearbeitung eines übermittelten Dokuments.
- Die Überwachung der MS Office Anwendung während der Bearbeitung eines Dokuments.

Weitere Komponenten des AutoUpdates sind:

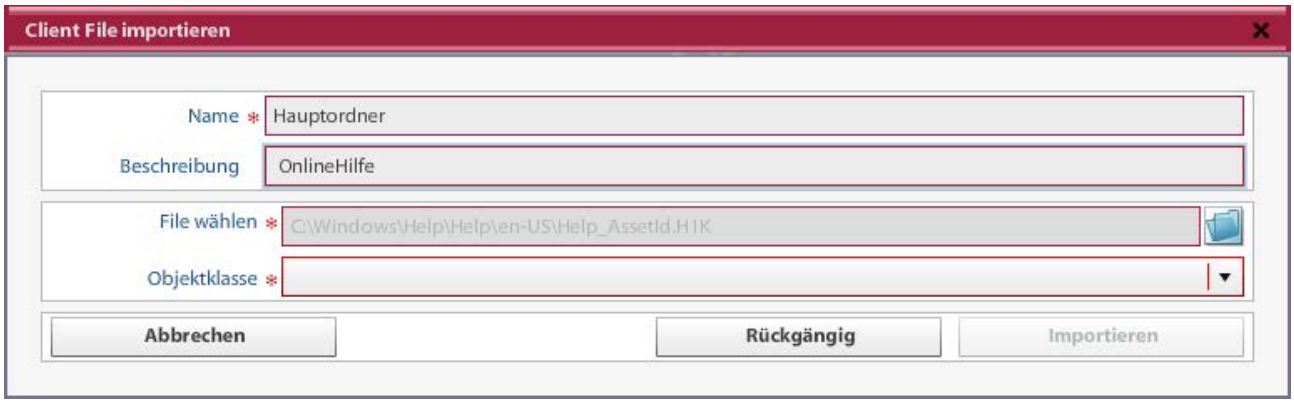
Datei	Beschreibung	Verzeichnis
iengine.dll	*i-engine* Standardbibliothek Zugriff auf Windows API (Registry, Environement)	%INSTALL_DIR%
olmailto.exe	Mailto für MS Outlook (Unterstützung von Anhänge)	%INSTALL_DIR%
outlooksync.dll	*i-engine* Integration für Outlook zur Task-Synchronisierung	%INSTALL_DIR%
para-docs_hdr.dot	*i-engine* Integration für Word (Header)	%WORD_STARTUP%
msmonitor.exe	Tool zum Überwachen der geöffneten Dateien	%INSTALL_DIR%
xcal.dat	Tasks von *i-engine* (Daten)	%TEMP%
i-engine.version	AutoUpdate Protokolldatei	
Optionale Komponenten		
gwmailto.exe	Mailto für Novell Groupwise (Unterstützung von Anhänge)	%INSTALL_DIR%

Client Files definieren

Die Liste der Dateien, welche beim AutoUpdate berücksichtigt werden sollen, ist beliebig erweiterbar. Die Steuerung, welche Anwender die entsprechenden Dateien erhalten sollen, wird in der Applikation über die «ClientFiles» durchgeführt.

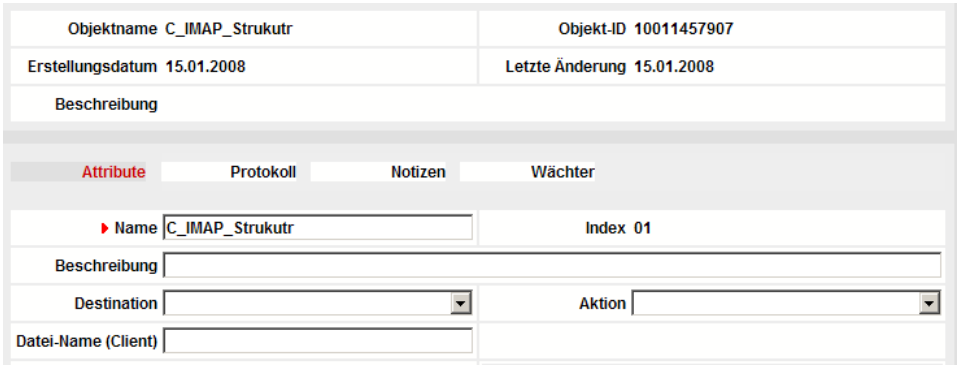
Innerhalb der Applikation wird auf jedem «ClientFile» Objekt den berechtigten Anwender-Gruppen der Zugriff erteilt. Beim Einloggen installiert *i-engine* dann die Komponenten Anwender spezifisch.

Alle ClientFiles werden in einem Ordner verwaltet. Soll eine weitere Datei integriert werden, muss auf dem entsprechenden Ordner die Aktivität **ClientFile importieren** gewählt werden.



Feld	Beschreibung
Name	Identifikation; Es wird automatisch der Dateiname aufgesetzt
Beschreibung	Ergänzende Informationen
File wählen	Datei
Objektklasse	Objektklasse wird entsprechend dem Dateityp automatisch aufgesetzt.
Importieren	Übernahme der Datei in *i-engine*

Im nächsten Schritt werden über das Kontextmenü **Eigenschaften bearbeiten**, die Parameter aufgesetzt.

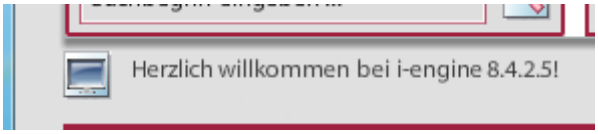


Feld	Beschreibung
Destination	Festlegen des Ablageortes auf dem Client-System Zwar kann hier ein fester Pfadname vorgegeben werden. Besser ist aber die Verwendung von Variablen, welche die aktuelle Einstellung des Clients berücksichtigen. «%TEMP%»: Temporär Verzeichnis «%EXCEL_STARTUP%»: Excel Programmverzeichnis «%EXCEL_ADDIN%»: Verzeichnis für Excel Erweiterungen «%WORD_STARTUP%»: WinWord Programmverzeichnis «%INSTALL_DIR%»: Generelles Programmverzeichnis «%INSTALL_USER_DIR%»: Individuelles Programmverzeichnis
Aktion	Festlegen der Behandlung auf dem Client-System. «COPY»: Datei auf Client kopieren «COPY, REGISTER»: Datei kopieren und registrieren «COPY,EXECUTE»: Datei kopieren und ausführen «SETENV»: Stellt das dynam. Verzeichnis als Umgebungsvariable dar. «DELETE»: Datei auf Client löschen «EXECUTE»: Datei auf dem Client ausführen «REGISTER»: Datei registrieren (regsrv32 notwendig) «REGEDIT»: Eintrag in Registry vornehmen (Dateityp ".reg") «UNREGISTER»: Datei un-registrieren. «UNREGISTER,DELETE»: Datei un-registrieren und löschen.
Dateiname (Client)	Die Datei kann auf dem Client-System einen vom Originalnamen abweichende Bezeichnung erhalten. Leer: gleicher Dateiname

Individualisierung

Message Ticker

Im Message Ticker wird standardmässig die aktuelle Versionsnummer der *i-engine* angezeigt.



DB Action

Die Meldungen des Message Tickers können in der Tabelle PROPERTIES angepasst werden. Es können mehrere Textzeilen definiert werden, die dem Benutzer alternierend angezeigt werden. Dabei sollte die erste Meldung nicht überschrieben werden, da diese von TIE bei jedem Software-Update überschrieben wird.

Feld	Eingabe
USER_NAME	Fix: «SYSTEM»
KEY	«MSG_0»: Standard System Meldung – besser nicht überschreiben «MSG_1»… : weitere Meldungstexte
VALUE _L1, _L2,_L3	Meldungstexte

Logo Link

In der rechten Ecke wird das Kundenlogo angezeigt. Darunter kann ein Link hinterlegt werden, z.B. auf die Intranetseite des Unternehmens.

Die entsprechende URL wird in der Tabelle PROPERTIES festgelegt.

Feld	Eingabe
USER_NAME	Fix: «SYSTEM»
KEY	«LOGO_URL»
VALUE _L1, _L2,_L3	URL Aufruf z.B. «http://www.bbi.ch»

Zeitplanen von Aufgaben

Um wiederkehrende Prozess regelmässig zu aktivieren, wird der Prozess-Scheduler von Oracle eingesetzt. Die eigentliche Aufgabe wird z.B. als PL/SQL Prozedur oder als Java-Script implementiert. Diese Aufgabe wird in der JOB Tabelle von Oracle eingetragen.

Um Einträge in der Aufgabenliste von Oracle vorzunehmen wird *p_create_job()* verwendet. Dazu wird ein Script erstellt, welcher die Parameter der Funktion aufsetzt und die Funktion aufruft. Für das Ausführen des Scripts ist eine Anmeldung als Schema-User notwendig.

Beispiel: Eintragen der Textindexierung

```
declare
      job_id_v varchar2(100) := 'J_TEXTINDEX';
begin
      p_drop_job (job_id_v);
      p_create_job (
      job_p        => job_id_v,
      what_p       => 'JOB_TEXTINDEX;',
      start_date_p => sysdate,
      interval_p   => 'FREQ=HOURLY; BYMINUTE=15,45',
      priority_p   => 3
      );
      commit;
end;
```

Standard Dienste

Die folgenden Dienste sind standardmässig für einen regelmässigen Lauf eingeplant:

- Watcher(): arbeitet die Tabelle WATCH_MAIL ab, und verschickt die eingetragenen Benachrichtigungen
- WatcherSystem(): arbeitet in der Tabelle SQL_STMT alle Einträge des Typs «SYSTEM_WATCHER» ab.

Weiter typische eingeplante Aufgaben sind:

- Fork Abarbeitung
- Datenoptimierungen

Geplanten Job löschen

Mit der Routine *p_drop_job(job_id)* wird ein eingeplanter Job wieder ausgetragen.

Die Ablage

Organisation der Datenablage

Die von *i-engine* verwalteten Dateien müssen physisch auf Festplatten eines als Datenserver agierenden Rechners abgelegt werden. Auf dem Datenserver übernimmt entweder eine spezielle FTP-Server Software oder ein in der Oracle Datenbank integrierter FTP-Dienst die Verwaltung der Dateien. Der FTP-Server/Dienst liefert die verlangten Dateien oder speichert sie auf der Festplatte ab. Die Ablage der Dateien in der Oracle Datenbank ist für die *i-engine* Anwendung transparent.

Der physische Ort der Datenablage ist unabhängig von der Struktur der *i-engine* Ordner. Damit ist eine Optimierung des vorhandenen Speicherplatzes bezüglich Auslastung und Sicherheit sehr einfach möglich.

Die Ablage wird üblicherweise in drei Bereiche strukturiert:

Systemablage	für alle *i-engine* internen Datenbestände
Kundenstammdaten	für Vorlagen, Konfigurationsdaten etc.
Arbeitsdaten	für alle anderen kundespezifischen Dateien
Archivablage	für langfristig gespeicherte Dateien, ggf. auch mehrere z.B. nach Restore-Zeit, Hardware-System,..
DICOM	Spezialablage für DICOM-Dateien

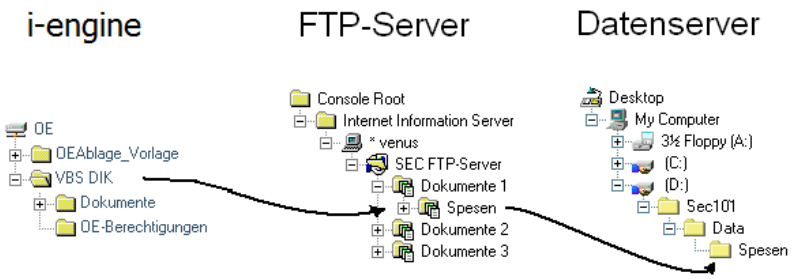
Hinweise

- Jedem Hauptordner kann individuell eine beliebige Ablage – und damit Speicherort – zugewiesen werden.
- Wenn mehrere FTP-Server verwendet werden, kann durch geschicktes Aufteilen der Ablage eine gleichmässige Auslastung der Datenträger erreicht werden.
- Bei lokal verteilten Systemen können die Daten auch auf «nahe» FTP-Server verteilt werden, so dass der Netzwerkverkehr minimiert werden kann.
- Jede in *i-engine* erzeugte Ablage bedingt auch eine entsprechende Anpassung bei den vom FTP-Server verwalteten Verzeichnissen resp. den Einstellungen in der Oracle Datenbank.

Der FTP-Server

Wo werden die Dokumente eigentlich gespeichert? Hier der Zusammenhang zwischen

- der Ordnerstruktur in *i-engine*,
- den virtuellen Verzeichnissen im FTP-Server und
- dem physikalischen Speicherort auf dem Datenserver



Eine neue Dateiablage

Beim Erstellen eines neuen Hauptordners kann jeweils die gewünschte Ablage für die Dokumente zugeordnet werden.

Variante mit FTP-Server

Um eine neue Ablage verfügbar zu machen sind folgende Arbeiten notwendig:

- Erstellen des physischen Verzeichnisses auf dem Datenserver.
- Erstellen des virtuellen Verzeichnisses im FTP-Server, welcher auf das physische Verzeichnis zeigt. s. Installationsanleitung FTP-Server.
- Erstellen eines Ablage Eintrages in der *i-engine* Konfiguration.
- Zuweisung der Ablage zum Hauptordner bei der Erstellung des Ordners.

Variante mit Oracle als FTP-Server

Die Dateien können alternativ direkt in der Oracle Datenbank abgespeichert werden. Um den Oracle-Server vorzubereiten, sind folgende Schritte durchzuführen:

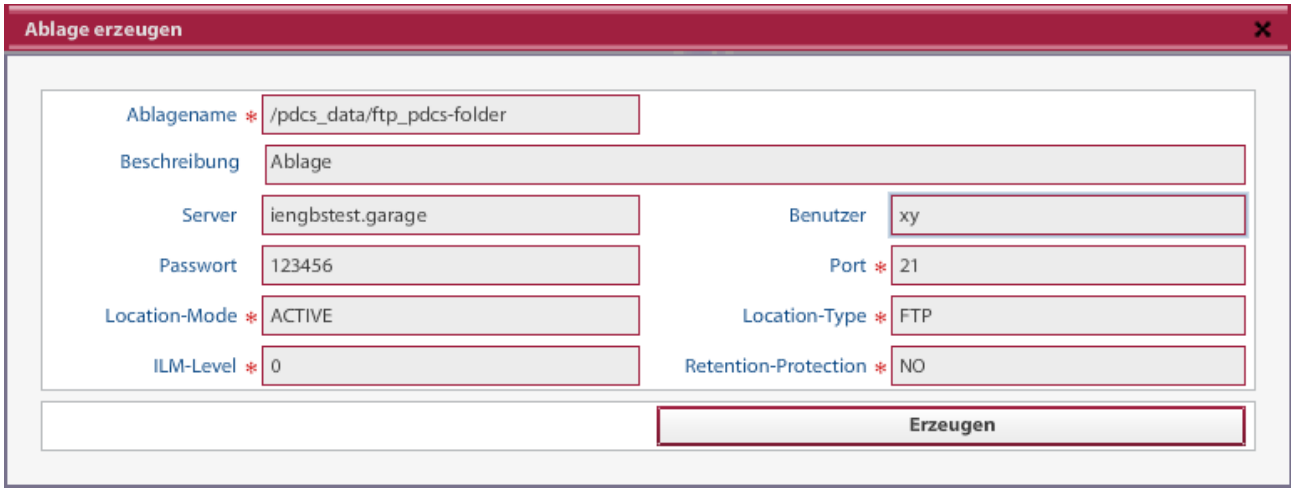
- XDB aufsetzen: In Oracle wird die XML-Komponente benötigt. Ab Oracle 11g sind speziell auch die Rechte auf die Verzeichnisse zu vergeben. d.h. ACL erzeugen, Rolle (DB-Schema) und Clients zuordnen.
- Der integrierte FTP-Server wird konfiguriert: z.B. Port definieren
- Erstellen eine virtuellen Verzeichnisses im FTP-Server

- Erstellen eines Ablage Eintrages in der *i-engine* Konfiguration.
- Zuweisung der Ablage zum Hauptordner bei der Erstellung des Ordners.

Erstellen eines Ablage Eintrages

Um eine neue Ablage zu parametrisieren gehen Sie wie folgt vor:

Unter «Basismodule > Ablagen» wird mit dem Kontextmenü ABLAGE ERZEUGEN die Eingabemaske aufgerufen.



Feld	Beschreibung
Ablagename	Die Bezeichnung **muss** dem Namen des virtuellen Verzeichnisses auf dem FTP-Server entsprechen. D.h. Pfad relativ zur FTP-Root.
Beschreibung	Ergänzende Beschreibung.
Server	DNS Name des FTP-Servers oder IP-Nummer.
Benutzer	Domain Benutzereintrag, welcher für den Zugriff auf das virtuelle Verzeichnis verwendet werden soll. Dieser Eintrag muss mit dem Benutzer übereinstimmen, welcher im FTP-Server angegeben wurde.
Passwort	Zugehöriges Passwort.
Port	Defaultwert für ftp ist «21»
Location-Mode	nur bei FTP: Kommunikationstyp «ACTIVE» / «PASSIV»
Location-Type	«FTP» «DB_DICOM» «SMB»: Windows Share «BLOB»: Binary Data «TSM»: Archivsystem

Feld	Beschreibung
	«URI»: URLbasierend
ILM-Level	nur bei TSM: Zugriffsgeschwindigkeit des Medium: tiefer Wert: schnell; hoher Wert: langsam
Retention-Protection	Schutz vor Überschreiben (zur Dokumentation) «NO» «YES»: bei TSM

Hinweis:

- Die Definition des virtuellen Verzeichnisses und FTP-Servers einer Ablage kann jederzeit abgeändert werden. Die Zuordnung der Ablage zum Hauptordner ist jedoch fix und kann später nicht mehr verändert werden.

Archivablage erstellen

Für die langfristige Archivierung von Daten werden spezielle Hardware-Systeme eingesetzt. Für jedes System muss ein zugehöriger Ablageeintrag erstellt werden. Die entsprechenden Einträge werden durch das Service/Operation Team von TIE erstellt.

DICOM-Ablage erstellen

DICOM Daten werden in einem separaten Teil der Oracle-Datenbank abgelegt und für den Zugriff zur Verfügung gestellt. Diese Datenbank muss in einem separaten Ablageeintrag erfasst werden.

Ablage zuordnen

Die Ablage muss dem Ordner-Objekt zugewiesen werden.

- Jedes Objekt enthält im Feld *Location* seinen Ablageort
- Beim Erzeugen eines Objektes in einem Container-Objekt vererbt sich der Ablageort des Containers auf das neue Objekt.
- Das nachträgliche Verschieben eines Container-Objektes von einem Ablageort in einen anderen hat keinerlei Auswirkungen auf die darin enthaltenen Objekte. d.h. sie bleiben wo sie sind. Praktischerweise kann deshalb der Ablageort nur für leere Container-Objekte aufgesetzt werden.

DB Action

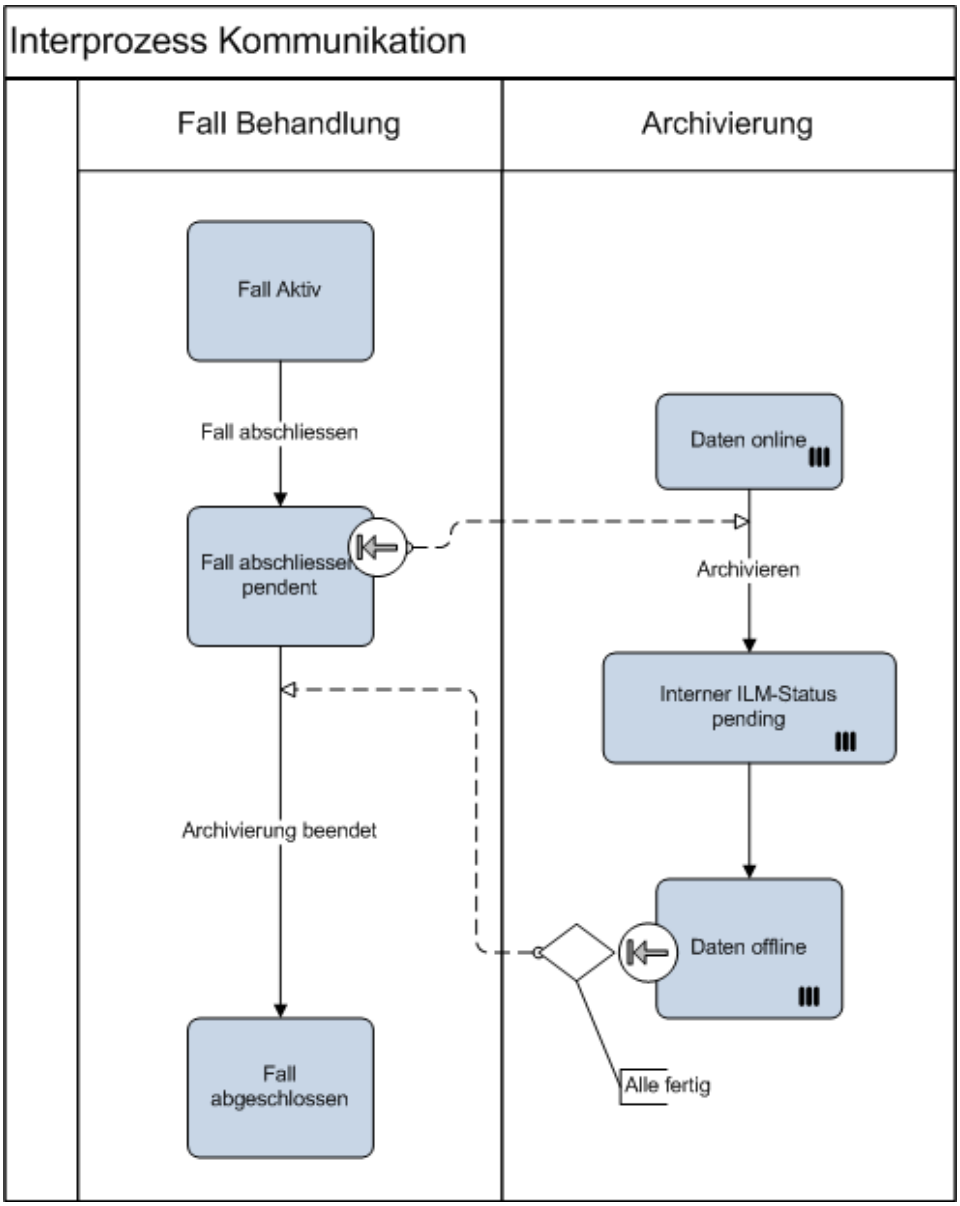
Die Zuordnung des Ablageortes zu einem Container-Objekt erfolgt direkt in der Tabelle OBJECT_TBL.

1. Vorgängig muss der Objekt-ID des Ablageortes bestimmt werden
2. In der Tabelle OBJECT_TBL muss der Eintrag des Container-Objektes bearbeitet werden. Im Feld *Location* wird die Objekt-ID der betreffenden Ablage eingetragen.

IPC - Interprozess Kommunikation

Abhängig von einem Ereignis im aktuellen Geschäftsprozess kann eine Aktivität in einem anderen Prozess ausgelöst werden. Damit lassen sich Prozesse synchronisieren oder der Ablauf eines zweiten Prozess wird durch einen ersten Prozess kontrolliert.

Beispiel einer Fall-Archivierung - asynchron



DB Action

Die Interprozess Kommunikation wird in der IPC Tabelle konfiguriert.

Prinzipiell besteht die Definition aus zwei Teilen:

- der Trigger-Bedingung, welcher die Voraussetzungen im 1ten Prozess für die Aktion definiert

und

- der Trigger-Aktion, welche die durchzuführende Aktion im 2ten Prozess festlegt.

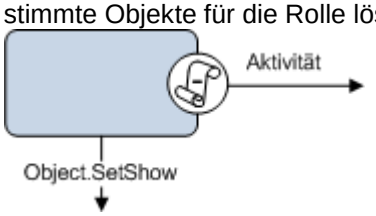
Feld	Eintrag
IPC_ID	Objekt ID
IPC_NAME	Freie Bezeichnung des IPCs Falls der IPC Namen nicht eindeutig ist oder die maximale Länge von 9 Zeichen überschritten wird, wird das IPC Mnemonic dem effektiven Namen vorangestellt.
EVENT	Trigger-Bedingung Ereignistyp «ON_ACTIVITY_START»: Bevor die Aktivität ausgeführt wird «ON_ACTIVITY_END»: Nachdem die Aktivität ausgeführt wurde «ON_STATUS_ENTRY»: Das Objekt erhält den Prozessstatus «ON_STATUS_EXIT»: Das Objekt verlässt den Prozessstatus «ON_EXPIRATION»: bei Überschreiten der Ablaufdauer (*) «ON_NOTIFICATION»: bei Überschreiten der Benachrichtigungsdauer (*) «ON_OBJECT_SETSHOW_REMOVE_ROLE»: wenn eine zugewiesene Rolle entfernt wird s. Hinweis (*) Diese sind im Prozessstatus definiert.
STATUS_ID resp. ACTIVITY_ID	Für die Trigger-Bedingungen «ON_ACTIVITY_START» und «ON_ACTIVITY_END» wird die ID der Aktivität angegeben, sonst die ID des Prozessstatus.
CONDITION_ STMT_ID	SQL-Statement welches prüft, ob der Trigger aktiv oder inaktiv ist. «YES» oder «»: Trigger verwenden «NO»: Trigger nicht verwenden
TARGET	Objekt auf welches die Trigger-Aktion ausgeführt werden soll. «stmt_id»: SQL-Statement liefert Liste der Zielobjekte; 1 Spalte mit allen Obj-IDs. «obj_id»: spezifische Objekt ; wenn die ID nicht als Statement-ID erkannt wird, wird eine Objekt-ID angenommen.

Feld	Eintrag
	«obj»: Zielobjekt ist das aktuelle Objekt selbst «parent»: Zielobjekt ist das übergeordnete Objekt «root»: Zielobjekt ist das nächste übergeordnete Root-Objekt «children»: Zielobjekte sind die direkt untergeordneten Objekte, dh. keine Objekte der untergeordneten Objekte. «branch»: Zielobjekte ist der gesamte untergeordnete Objektbaum, dh. auch alle Objekte der untergeordneten Objekte, und deren untergeordnete Objekte. Auch eine Verkettung von «parent» und «root» Angaben ist möglich. z.B. «parent.root.parent»
TARGET_ ACTIVITY_ID	Aktivität, die auf dem/den Zielobjekt(en) ausgeführt wird. oder
TARGET_ METHODE_ID TARGET_ REFERENCE	Methode - ggf. zusammen mit TARGET_REFERENCE), um Eindeutigkeit zu erreichen - welche auf dem/den Zielobjekt(en) ausgeführt wird. Wird verwendet, wenn TARGET_ACTIVITY_ID nicht aufgesetzt ist. Diese Variante wird im Zusammenhang mit mehreren Targets eingesetzt, wenn ggf. unterschiedliche Methoden erforderlich sind. **Hinweis**: Die Spezifikation mit einer Methode ist jener mit einer Aktivität vorzuziehen, da keine applikationsspezifischen IDs verwendet werden.
TARGET_ STATUS_ID	Um unnötige Protokolleinträge zu vermeiden, kann das Auslösen des IPCs auf einen bestimmten Prozessstatus beschränkt werden. Wenn eine Status-ID festgelegt wird, erfolgt eine Überprüfung der IPC-Bedingungen nur noch, wenn sich das Target-Objekt in diesem Prozessstatus befindet.
TARGET_ PARAMS	Für diesen IPC-Eintrag spezifische Parameter, mit welchen die Aktivität ausgeführt wird. Syntax gemäss *i-engine V8* z.B. wf_note=Stellungnahme&template_id=12345678&sys_attribute.2.t1=Patient.
DUE_DATE	Festlegen, wann die Trigger-Aktion durchgeführt werden soll. «Null»: sofort, synchron «sysdate»: sofort, asynchron «sysdate+n»: Anzahl Tage ab heute; Bsp. 1.5 => 36h
SEQ	Abarbeitungsreihenfolge, wenn mehrere Trigger für die gleiche Situation definiert sind.
REPEAT	Anzahl Wiederholungen

Feld	Eintrag
	Der IPC kann in bestimmten Zeitintervallen wiederholt werden, solange seine Auslösebedingungen erfüllt sind. Das Zeitintervall ist durch den Wert in DUE_DATE vorgegeben.

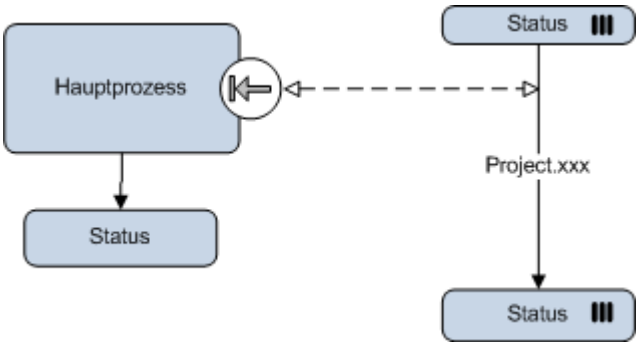
Hinweis

- Die Verarbeitung eines IPC-Aufrufes kann länger dauern. Während der Bearbeitung befinden sich die Objekte in einem vordefinierten Prozessstatus. Diese Status sind vorkonfiguriert und werden im Workflow automatisch eingesetzt.
  Die entsprechenden Status sind unter *Basismodule* > *Prozess* > *IML Zwischenstatius* abgelegt.
- Der Einsatz der IPC-Funktionalität bedeutet immer, dass die Parameter für die Methode, welche in der ausgelösten Aktivität verwendet wird, durch ein SQL-Statement aufgesetzt werden müssen. Als Vorlage für das Parameter-Statement ist jeweils die Methoden-spezifische PL/SQL-Funktion CF_PAR_xxxx einzusetzen.
- Wird über die Methode *setShow()* die Rollenzuweisung zu einem Objekt gelöscht, wird das Ereignis ON_OBJECT_SETSHOW_REMOVE_ROLE ausgelöst. Es kann damit bspw. eine Aktivität gestartet werden, welche bestimmte Objekte für die Rolle löscht.



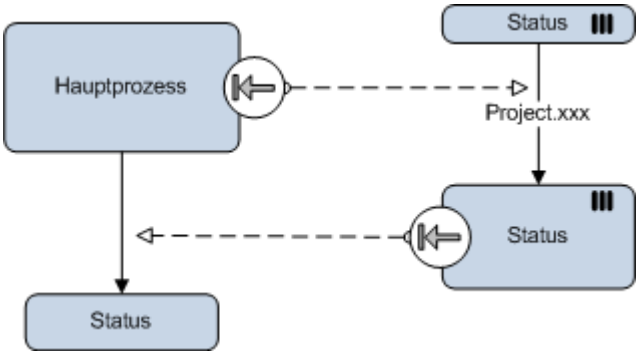
Synchron/Asynchrone Aktivitäten

Bei einer synchronen Abarbeitung der IPC wird der Hauptprozess - d.h. die auslösende Aktivität - erst fortgesetzt, wenn die gestartete Aktivität komplett ausgeführt wurde.



- sehr einfach Implementierung einer synchronen Verarbeitung im Workflow, insbesondere bei der Verwendung von Projekt-Methoden.
- ggf. längere Blockierung des Hauptprozesses, ohne Feedback-Möglichkeit für den Benutzer. Auch keine Möglichkeit einen Abbruch durch den Benutzer herbei zu führen.
- Eine Synchronisierte Ausführung ist nur innerhalb des Ablaufes des Geschäftsprozesses sinnvoll. Ereignisse die in sich bereits asynchron ablaufen - bspw. Aktivität bei Ablauf-/Mitteilungsdatum und Rollenänderung - darf keine synchrone Verarbeitung eingesetzt werden.

Die IPC-Kommunikation ermöglicht auch das asynchrone Ausführen von Aktivitäten. D.h. für den Benutzer bleibt die Maske nach Auslösen der Aktivität nicht blockiert, sondern er kann sofort weitere Aufgaben ausführen. Im Hintergrund wird die ausgelöste Aktivität zu Ende gebracht.



- aufwändigere Implementation, falls eine Rückmeldung an den Hauptprozess notwendig ist.

- der angestossene Prozess kann auch mehrere Prozessschritte bis zu einer Rückmeldung an den Hauptprozess durchlaufen.
- die Rückmeldung kann von Bedingungen abhängig gemacht werden.

Sicherheitsklassifikation

DB Action

Durch die Sicherheitsklassifikation lassen sich Objekte speziell schützen. Die eigentlichen Objektrechte (Zugriff und Sichtbarkeit) sind unabhängig von der Sicherheitsklassifikation, werden aber meistens entsprechend der Sicherheitsklassifikation aufgesetzt.

Die Sicherheitsklassifikation eines Objektes bedeutet:

- eingeschränkte Nutzung des Objektes
- spezielle visuelle Markierungen
- Verschlüsselung der Dateninhalte

Die Sicherheitsklassifikation wird insbesondere in folgenden Situationen eingesetzt:

- Sicherheitsklassifizierte Rollen werden nicht an den Stellvertreter weitergegeben. Dies ist z.B. bei der «Privâte» Rolle so.
- Die Anzeige und Verarbeitung von sicherheitsklassifizierten Objekten kann speziell gesteuert werden. z.B. unterschiedliches Symbol für solche Objekte, zusätzliche Information bei Ausgaben («intern», «geheim»)
  Symbol für nicht klassifizierte Rolle
  Symbol für sicherheitsklassifizierte Rolle
- Bei Dokumenten wird der Inhalt automatisch verschlüsselt abgelegt.

Festlegen der Sicherheitsklassifikation

Die Funktionen müssen dem Benutzer bei Bedarf durch das Customizing speziell zur Verfügung gestellt werden. Die Sicherheitsklassifikation von Rollen ist standardmässig in *i-engine* implementiert. Die Informationen zur Klassifikation sind in der Tabelle PROPERTIES abgelegt.

Alle Einträge sind unter *User_name*=«SYSTEM» abgelegt. Für jede Sicherheitsstufe gibt es zwei Einträge, welche im Feld *Key* gekennzeichnet sind.

Der Eintrag ENCRYPTION

Feld	Bezeichnung
USER_NAME	«SYSTEM» fix
KEY	«ENCRYPTION» fix
SEQ	Sicherheitsstufe Bei Werten über 100 wird das Symbol automatisch mit einem eingeblendeten Schüssel ergänzt. Dem Benutzer werden die gesicherten Dokumente zudem via erweitertem

Feld	Bezeichnung
	ToolTip Text auf der Objektklasse visuell markiert. Bei einem Wert >0 wird im Tooltip zusätzlich der Text «Klassifiziert» angezeigt. «1»: keine Sicherheitsstufe «301»: sicherheitsklassifizierte Rolle «101»: vertraulich
VALUE VALUE_L1,L2,L3	Verschlüsselungscode Für die Verschlüsselung von Dokumenten kann der Verschlüsselungscode hinterlegt werden. Achtung: Eine Änderung macht alle bereits verschlüsselten Dokumente nicht mehr lesbar. «NONE»: Keine Verschlüsselung verwenden «hhHx8M*,»: Alphanumerischer Schüsselcode Da die Tabelle abhängig von der Spracheinstellung das Wertefeld zurückliefert, muss der Wert in allen 4 aufgesetzt werden.

Der Eintrag ENCRYPTION_RESTRICTION

Feld	Bezeichnung
USER_NAME	«SYSTEM» fix
KEY	«ENCRYPTION_RESTRICTION» fix
SEQ	Sicherheitsstufe
VALUE VALUE_L1,L2,L3	Dokument verschlüsseln: Wenn die Methode *Object.Classify()* ausgelöst wird, kann der Benutzer u.U. entscheiden, ob er die Datei verschlüsseln will. «NONE»: Keine Verschlüsselung verwenden «OPTIONAL»: Dem Benutzer wird eine Dialogmaske angezeigt, wo er entscheiden kann, ob er die Datei verschlüsseln will. «MANDATORY»: Die Datei wird immer verschlüsselt. Da die Tabelle abhängig von der Spracheinstellung das Wertefeld zurückliefert, muss der Wert in allen 4 aufgesetzt werden.

Hinweis:

- Natürlich macht es nur Sinn in «ENCRYPTION_RESTRICTION» eine Verschlüsselung zu fordern, wenn auch ein Schüsselcode für diese Sicherheitsstufe festgelegt wurde.
- In der angezeigten Dialogmaske wird üblicherweise auch eine Einschränkung der Dateibenutzer festgelegt, so dass nur noch ausgewählte Benutzer die entsprechenden Objektrechte besitzen.

Benachrichtigung Konfigurieren

DB Action

Die Benachrichtigung kann als E-Mail oder SMS-Meldung erfolgen. Die Übermittlung muss vorgängig aufgesetzt werden.

E-Mail Server Grundeinstellungen

Für das Versenden von E-Mails ist ein SMTP-Server notwendig. Dazu sind in der Tabelle PROPERTIES folgende Einträge notwendig:

USER_NAME	«SYSTEM»
KEY / VALUE	«MAILER»: E-Mail Typ – unterstützt sind: «olmailto» Outlook «gwmailto» Groupwise
	«MAILHOST»: Adresse des Mail-Servers
	«WATCHERCONTENT_TYPE»: Das System ist standardmässige zur Versendung der E-Mails im Text-Format aufgesetzt (MIME: Text/plain). Diese Einstellung durch einen Eintrag in dieser Tabelle geändert werden: «HTML» Es wird MIME Text/html verwendet. Diese Einstellung ist nur für E-Mails relevant, die nicht durch eine Vorlage erzeugt werden (Systemüberwachung).

Hinweis

- Der Mail-Server muss Meldungen ohne Authentifizierung akzeptieren.
- Es ist zu beachten, dass E-Mails im HTML-Format typischerweise erweiterte Inhalte benötigen.

SMS Grundeinstellungen

Für den Versand von SMS-Nachrichten sind in der Tabelle PROPERTIES folgende Einträge notwendig:

USER_NAME	«SYSTEM»
KEY / VALUE	«SMS_ADDRESS»: E-Mail-Adresse für SMS-Empfang
	«SMS_USERNAME»: Angaben für die Anmeldung am SMS-System

	«SMS_PASSWORD»
	«SMS_FROM»: Absender der in der SMS-Meldung angezeigt wird.

Projekt Methoden

Über Projekt Methoden können Methoden in allen Objekten ausgelöst werden, welche dem aktuellen Objekt untergeordnet sind. Dadurch können sehr effiziente Verarbeitungen für ganze Objektgruppen implementiert werden.

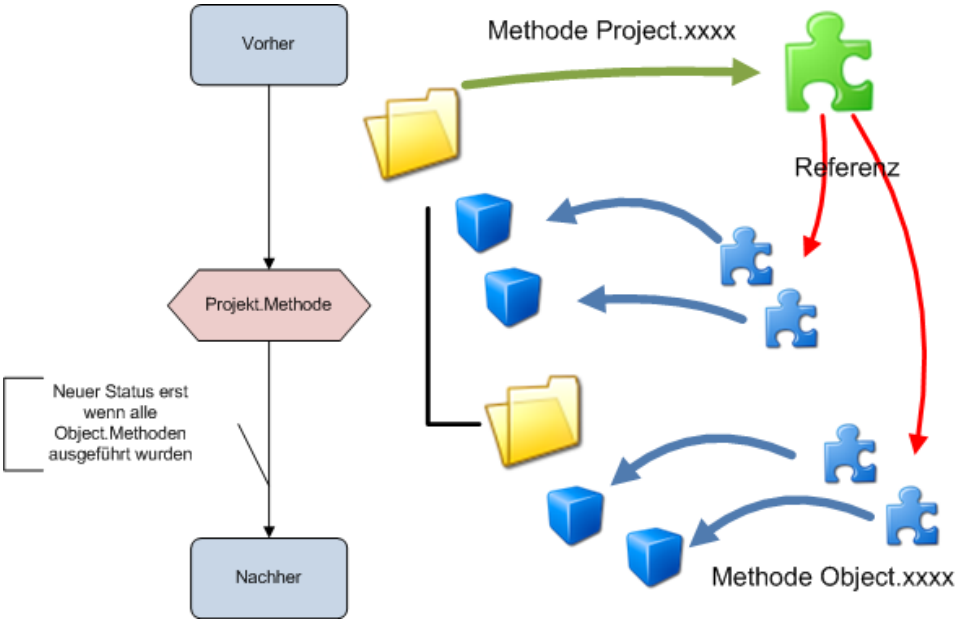
Projekt Methoden anwenden

Um gleichzeitig eine Methode auf mehrere Objekte auszuüben, stellt i-engine Projekt-Methoden zur Verfügung. Die Projekt-Methoden werden auf einem Container-Objekt ausgeführt und lösen in allen darin enthaltenen Objekten die zugehörigen Objekt-Methoden aus.

Für viele Objekt-Methoden sind entsprechende Projekt-Methoden verfügbar. Die Projekt-Methode stösst für alle untergeordneten Objekte die entsprechende Objekt-Methode an. Voraussetzung ist dabei, dass sich das untergeordnete Objekt in einem Prozesszustand befindet, der eine Aktivität mit dieser Objekt-Methode einsetzt.

Sollten mehrere Aktivitäten mit der gleichen Objekt-Methode in Frage kommen, muss eine Auswahl über einen Referenzbezeichner in der Projekt- und den Objekt-Aktivitäten vorgesehen werden.

Alle untergeordneten Objekte werden rekursiv abgearbeitet. Oft ist dies nicht gewünscht. Deshalb kann die Auswahl der Objekte durch ein SQL-Statement, welches in OBJECT_PROPERTY hinterlegt wird, auf die gewünschten eingeschränkt werden. Auch auf dem Hauptobjekt wird die Objekt-Methode ausgeführt. Falls ein SQL-Statement für die Filterung eingesetzt wird, muss dieses natürlich diese Bedingung ggf. auch berücksichtigen.



Jedes Objekt führt die angestossene Aktivität aus und nimmt ggf. auch den anschliessend vorgesehenen individuellen Prozessstatus ein. Verwendet die Objekt-Aktivität Parameter aus dem Attributprofil, so werden diese aus dem Attributprofil genommen, welches für die Projekt-Aktivität definiert wurde.

Für das Hauptobjekt sind die Aktivitätseinstellungen der Projekt-Aktivität zuständig.

Die Bedingungen für das Auslösen der Objekt-Methode in untergeordneten Objekten sind:

- Objekt befindet sich in einem Prozessstatus, welche eine Aktivität mit der entsprechenden Objekt-Methode hat.
- Der Benutzer hat die Berechtigung für das Ausführen der Aktivität.
- Die Referenzen in Projekt- und Objekt-Methode sind identisch.
- Das Objekt wurde ggf. durch das SQL-Statement selektiert.

Einschränkung der betroffenen Objekte

Oft ist es nicht gewünscht, dass die Methode bei allen untergeordneten Objekten ausgelöst wird. Abhängig von der Anwendung befinden sich z.B. auch Objekte im Container, für welche die Methode nicht angewendet werden soll, obwohl sie im aktuellen Objekt Prozessstatus verfügbar wäre.

- Es kann ein SQL-Statement definiert werden, welches die zu verarbeitenden Objekte liefert. Dieses wird in der OBJECT_PROPERTY Tabelle definiert.
- Es wird ein Listelement mit einer Auswahlbox verwendet. Nur ausgewählte Elemente werden berücksichtigt.

Oft ist es für den Benutzer nicht ersichtlich, welche Objekte durch die Projekt-Methode tatsächlich bearbeitet werden. Es kann deshalb erforderlich werden, dass ein Before-Checker die Konditionen prüft und ggf. eine Meldung angezeigt wird.

Liste der Projektfunktionen

Project.AddAlias	Object.AddAlias
Project.Attributes.Edit	Object.Attributes.Edit
Project.Attributes.View	Object.Attributes.View
Project.BulkAttributesEdit	Object.Attributes.Edit
Projekt.Checkin	Object.Checkin
Project.Checkout	Object.Checkout
Project.CheckoutDownload	Object.Checkout Object.Download2Temp
Project.ClientMail	Object.ClientMail
Project.CreateAlias	Object.CreateAlias
Project.Commit	Object.Commit
Project.Copy	Object.Copy
Project.Copy2Location	Object.Copy2Location
Project.Cut	Object.Cut
Project.Delete	Object.Delete
Project.DeleteAlias	Alias.Delete
Project.Distribute	Object.Distribute
Project.Download2Temp	Object.Download2Temp
Project.Edit	
Project.Export	
Project.NewLink	Object.NewLink
Project.NewVersion	Object.NewVersion
Project.MoveAliases2MaxVersion	Alias.Move2MaxVersion
Project.MoveLinks2MaxVersion	Link.GerMaxVersion
Project.Move2Location	Object.Move2Location
Project.RemoveFromLocation	Object.RemoveFromLocation

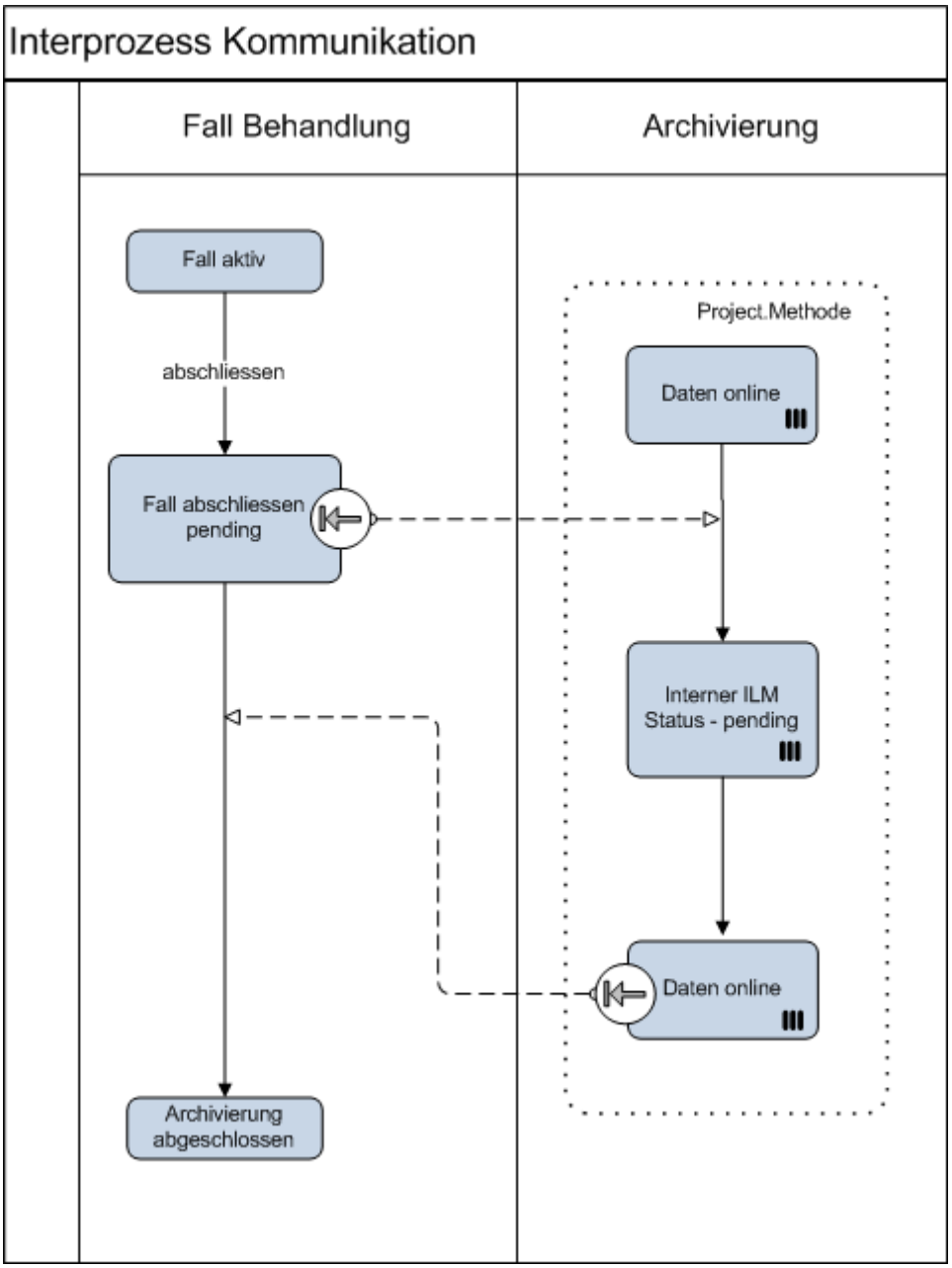
Project.RemoveLink	Object.RemoveLink
Project.Revert	Object.Revert
Project.Scrap	Object.Scrap
Project.SetAccess	Object.SetAccess
Project.SetShow	Object.SetShow
Project.Update	Object.Update

Nicht alle Methoden welche einen «Projekt» Namen tragen sind in diesem Sinn Projekt-Methoden. Auch gibt es nicht zu allen Objekt-Methoden die erforderlichen Projekt-Methoden. s. dazu nächsten Abschnitt.

Falls das Hauptobjekt ebenfalls eine Aktivität mit der Objekt-Methode aufweist, wird durch die Projekt-Methode auch diese angestossen. Der Übersichtlichkeithalber sollte in einem solchen Fall nur eine der beiden Methoden den «Nächsten Status» beeinflussen. Oft ist das Auslösen der Objekt-Methode auf dem Hauptobjekt unerwünscht und muss über die Verwendung von Referenzen unterbunden werden.

IPC und Projekt-Methoden kombinieren

Mit dem IPC-Mechanismus – s. «Customizing > Spezielle Konfigurationsaufgaben > Interprozess Kommunikation» - kann die Funktionalität und/oder Leistungsfähigkeit der Projekt-Methoden wesentlich erweitert werden.



Der Einsatz von Projektfunktionen und IPC bringt folgende Vorteile:

- Das Protokollieren der Projektfunktion geschieht erst am Ende der Aktivität.
- Die Synchronisierung auf das Aktivitätsende für alle beteiligten Objekte übernimmt die Projektfunktion selbst.
- Es werden weniger Jobs generiert, was zu besserer Systemleistung führt.

Konfigurationsbeispiele

Cut&Paste Operationen

Für die Implementierung von Cut&Paste resp. Copy&Paste Operationen stehen entsprechend den Anforderungen unterschiedliche Mechanismen zur Verfügung.

Objekt Transfer

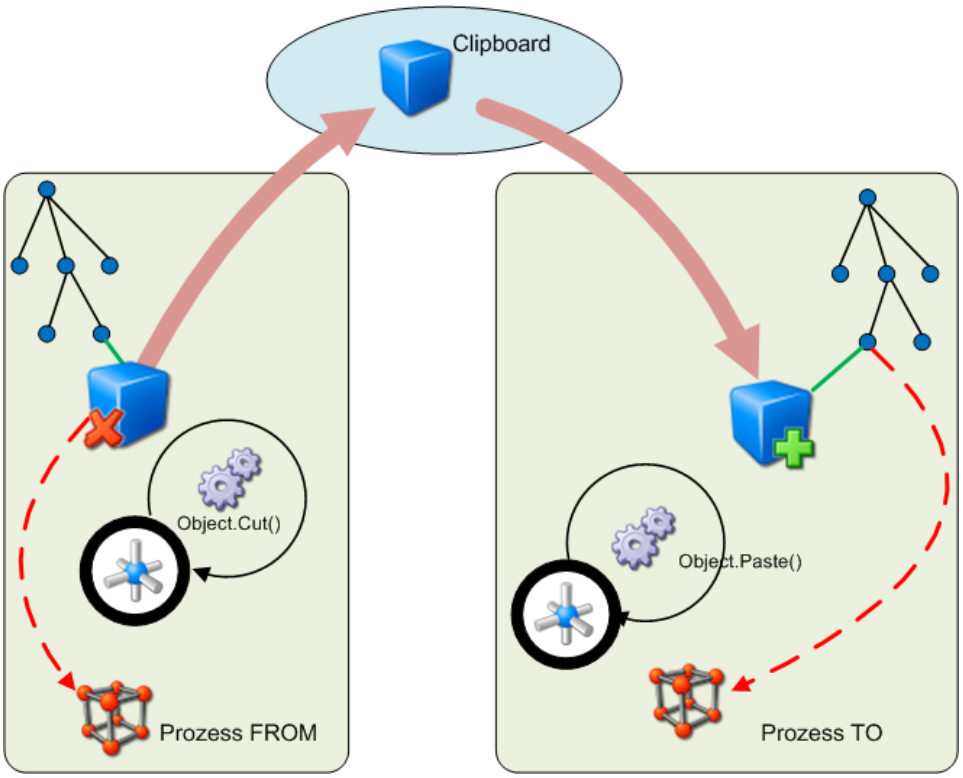
Vorgang:

Ein Objekt soll von einer Strukturposition an eine andere Position bewegt werden, ohne Änderungen an seinem Prozesszustand vorzunehmen.

Implementation:

- Die Methoden *Object.Cut()* steht auf dem betreffenden Objekt zur Verfügung.
- Im Zielordner der Struktur muss die Methode *Object.Paste()* für den Benutzer verfügbar sein.

- Beim Vorgang wird die Referenz im aktuellen Container gelöscht und dafür im neuen Container eingefügt. Ansonsten ändern sich beim Objekt nichts.



Hinweis

- Die Implementierung ist für *Object.NewLink()* und *Object.CreateAlias()* identisch.
- Allenfalls betroffene Dateien werden am Ablageort des Zielcontainers abgelegt. Dh. falls notwendig, werden die Dateien durch die Methode automatisch von einer Ablage in die andere verschoben.

Fixer Prozessstatus

Vorgang:

Die simple Variante lässt dahingehend erweitern, dass das Objekt an der neuen Strukturposition zusätzlich auch einen neuen Prozesszustand einnehmen soll. Allerdings ist der Status fix vorgegeben – entweder beim Ausschneiden/Kopieren oder beim Einfügen. Dh. der neue Status wird dem Objekt mitgegeben oder aber beim Einfügen in den Zielordner vorgegeben.

Implementation Verschieben:

- Die Methoden *Object.Cut()* steht auf dem betreffenden Objekt zur Verfügung. Möchte man dem Objekt den neuen Status mit auf den Weg geben so wird das Feld *Nächster Status* entsprechend aufgesetzt.
- Im Zielordner der Struktur muss die Methode *Object.Paste()* für den Benutzer verfügbar sein. Das Feld *Neuer Status* kann aufgesetzt werden und legt

dadurch den gewünschten Prozessstatus des Objektes an seinem neuen Platz fest. Sollte bereits beim *Object.Cut()* ein Status angeben worden sein, so wird dieser überschrieben.
Im Feld *Nächster Status* könnte dem Zielordner zudem ein neuer Status zu gewiesen werden, welcher z.B. weiter Paste-Operation verhindert.

Implementation Kopieren:

- Die Methoden *Object.Copy()* steht auf dem betreffenden Objekt zur Verfügung. Möchte man dem Objekt den neuen Status mit auf den Weg geben so wird das Feld *Neuer Status* entsprechend aufgesetzt.
  Im Feld *Nächster Status* könnte zudem für das Originalobjekt ein neuer Status festgelegt werden, welcher z.B. berücksichtigt, dass das Objekt nun kopiert wurde.
- Im Zielordner der Struktur muss die Methode *Object.Paste()* für den Benutzer verfügbar sein. Das Feld *Neuer Status* kann aufgesetzt werden und legt dadurch den gewünschten Prozessstatus des Objektes an seinem neuen Platz fest. Sollte bereits beim *Object.Status()* ein Status angeben worden sein, so wird dieser überschrieben.

Hinweise

Die Verwendung dieser erweiterten Variante hat wesentliche Nachteile:

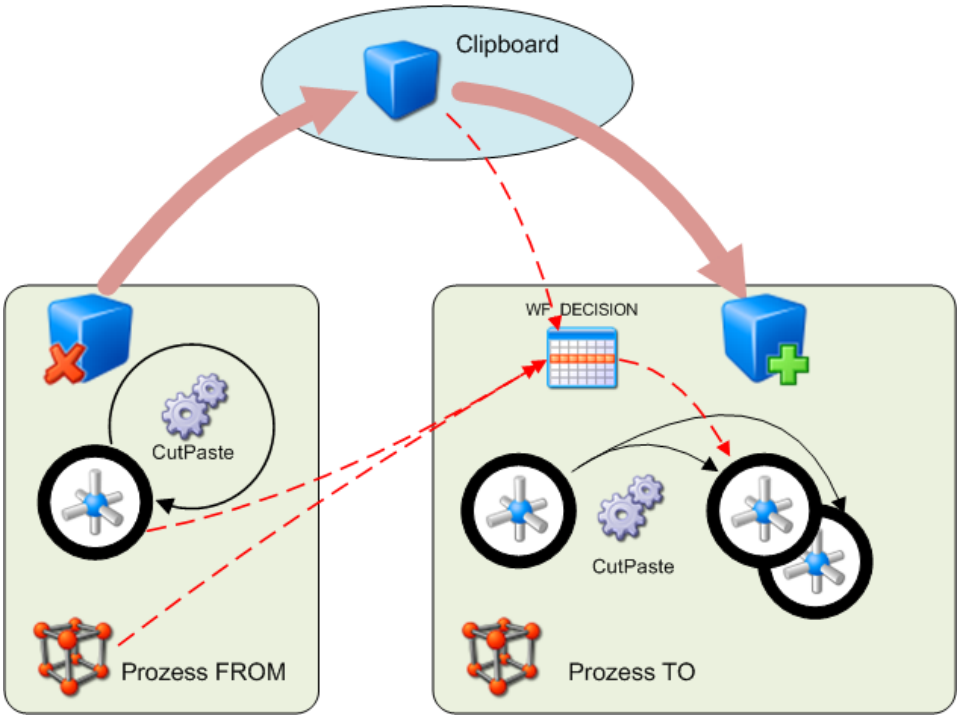
- alle Objekte vom resp. in den gleichen Ordner erhalten den gleichen Status. Eine Fallunterscheidung ist nicht möglich.
- Die *Object.Paste()* Operation kann nicht kontrollieren, ob es sinnvoll ist, das betreffende Objekt einzufügen. Unterschiedliche Cut&Paste Anwendungsfälle können vom Anwender ungewollt gemischt gebracht werden.

Es empfiehlt sich, diese Variante nur in speziellen Situationen einzusetzen.

Prozesswechsel

Vorgang

Bei einer Cut&Paste resp. Copy&Paste Operation wird ein Objekt aus einem Zustand im Prozess «FROM» in einen anderen Prozess «TO» umkopiert. Dieser Vorgang ist z.B. notwendig, wenn ein Dokument aus einem privaten Bereich in den öffentlichen Bereich verschoben wird. In beiden betroffenen Prozessen muss jeweils eine entsprechende Aktivität zur Verfügung stehen.



Es gilt die beiden Aktivitäten im Prozess «FROM» und Prozess «TO» richtig aufzusetzen.

Prozess «FROM»

- Für die Methode *Object.Cut()* resp. *Object.Copy()* muss im Feld *Nächster Status* festgelegt werden, welcher Prozessstatus nach einem erfolgreichen Paste-Operation eingenommen werden soll.

Prozess «TO»

- In der Tabelle WF_DECISION ist für jede Kombination von Ursprung Prozess/Prozessstatus und Ziel Prozess/Prozessstatus, welche die «Paste» Aktivität bearbeiten soll, ein Eintrag notwendig. Alle Einträge in dieser Tabelle beziehen sich demnach auf die Methode *Object.Paste()*.

Feld	Bedeutung
ACTVITY_ID	Aktivität mit der Methode *Object.Paste()* auf die sich der Eintrag bezieht.
SUB_METHODE_ NAME	Definition, welche «Paste» Aktivität mit diesem Eintrag behandelt wird. «Object.CopyPaste()» «Object.CutPaste()» sowie weitere
OBJ_TYPE	Festlegen, auf welches beteiligte Objekt sich der Eintrag bezieht und auf das die Aktion angewendet werden soll. «this»: für diesen Vorgang nicht verwendet «new» Bei «Object.CopyPaste()» muss ein neues Objekt erzeugt werden. «clipboard»: Bei «Object.CutPaste()» befindet sich das aktuelle Objekt im Clipboard.: s. weiter unten Abschnitt «Was ist welches Objekt?»
SEQ	Für die Aktivität können mehrere Paste-Operationen festgelegt werden. Oft werden z.B. abhängig von der Art des Objektes individuelle Operationen angeboten. Ein letzter Eintrag bietet eine generelle Paste-Operation für alle nicht speziell unterstützten Objekte. Über die Sequenz-Nummer wird die Reihenfolge bei der Bestimmung der Paste-Operation gesteuert.
FROM_PROCESS_ REF FROM_STATUS_REF	Definition des Urspung-Prozesses, wo der «Cut» resp. «Copy» Vorgang stattgefunden hat. Die Referenz Bezeichner sind in der jeweiligen Prozess- resp. Prozessstatus-Konfiguration festzulegen. «*»: das betreffende Feld wird für die Auswahl des Eintrages nicht geprüft. Üblicherweise wir FROM_PROCESS_REF auf «*» gesetzt, da der Status für die Auswahl genügt.
TO_PROCESS_REF TO_STATUS_REF	Definition des Ziel-Prozesses, wo der «Paste» Vorgang stattfindet. Die Referenz Bezeichner sind in der jeweiligen Prozess- resp. Prozessstatus-Konfiguration festzulegen. «*»: das betreffende Feld wird für die Auswahl des

Feld	Bedeutung
	Eintrages nicht geprüft. Üblicherweise wir TO_PROCESS_REF auf «*» gesetzt, da der Status für die Auswahl genügt.
SCOPE_LEVEL	Es kann festgelegt werden, innerhalb welchem Bereich nach TO_.. gesucht werden soll. «0»: Global -> empfohlene Einstellung «1»: bis zum nächsten übergeordneten Root-Element des Objektes. «2», «3»: entsprechend bis zu einem überordneten Root-Element.

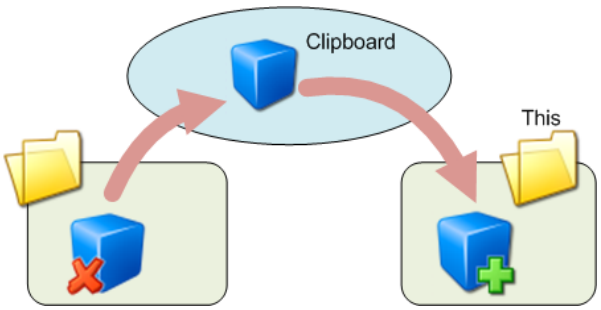
Hinweis

- Bei einer *Object.CopyPaste()* Operation werden üblicherweise 2 Einträge gemacht.
  OBJ_TYP=«new»: Festlegen was mit dem neuen Objekt geschieht.
  OBJ_TYP=«this»: Das bisherige Objekt bearbeiten, z.B. neuer Status oder Log-Eintrag.

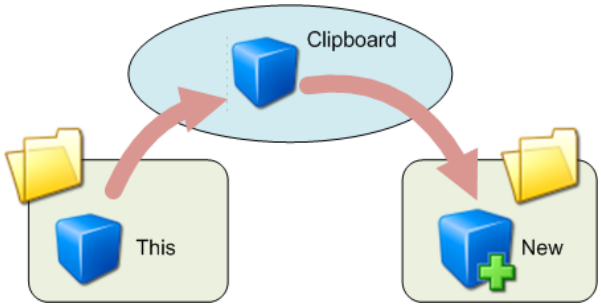
Was ist welches Objekt?

Im zeitlichen Ablauf des Vorgangs sind die symbolischen Objekte «this», «new» und «clipboard» vorgesehen. Die in WF_DECISION eingetragenen Aktivitäten werden jeweils für ein symbolisches Objekt definiert. Es ist deshalb wichtig zu wissen, welche Objekte damit eigentlich gemeint sind. Nun ist dies aber ist eine Wissenschaft für sich. Deshalb hier eine Zusammenfassung:

Cut&Paste Operation



Copy&Paste Operation



Projekt

Mit den Methoden *Project.Cut()* resp. *Process.Copy()* werden nicht einzelne Objekte, sondern alle im Projekt-Objekt enthaltenen Objekte durch die Cut&Paste Operation bearbeitet. Das Einfügen der Objekte wird – wie bei der *Object.Cut()* resp. *Object.Copy()* Methode – durch die Methode *Object.Paste()* ausgeführt.

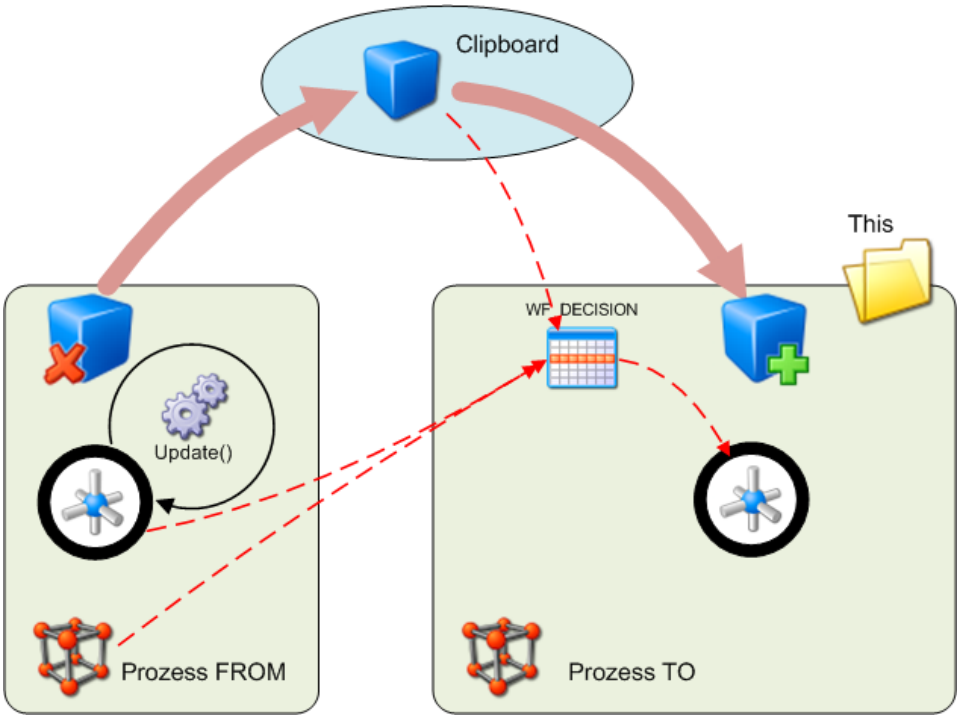
Object.Paste() reagiert unterschiedlich, abhängig davon, ob vorgängig eine Projekt oder Objekt-Methode eingesetzt wurde.

- Object: Es wird das normale Attributprofil der Aktivität verwendet
- Project: Es wird das Project-Attributprofil der Aktivität verwendet. Anschliessend werden für alle betroffenen Objekte nacheinander die *Object.Cut/Copy* und *Object.Paste()* Operationen durchgeführt. Bei jeder Operation werden die Eingaben aus dem Projekt-Attributprofil verwendet.

Mit diesem Mechanismus können unterschiedliche Parameter beim Cut&Paste Vorgang eines Einzelobjekte und von Mehrfachobjekten verwendet werden.

Dynamischer Prozesswechsel

Eine weitere Variante für die dynamische Prozesssteuerung ist der Wechsel eines Objektes von einem Prozess in einen anderen. Prinzipiell handelt es sich um eine degenerierte Variante der Cut&Paste-Operation mit Prozesswechsel, wie im vorherigen Abschnitt beschrieben. Allerdings ist in diesem Fall nur eine einzige Operation für den Vorgang verantwortlich.

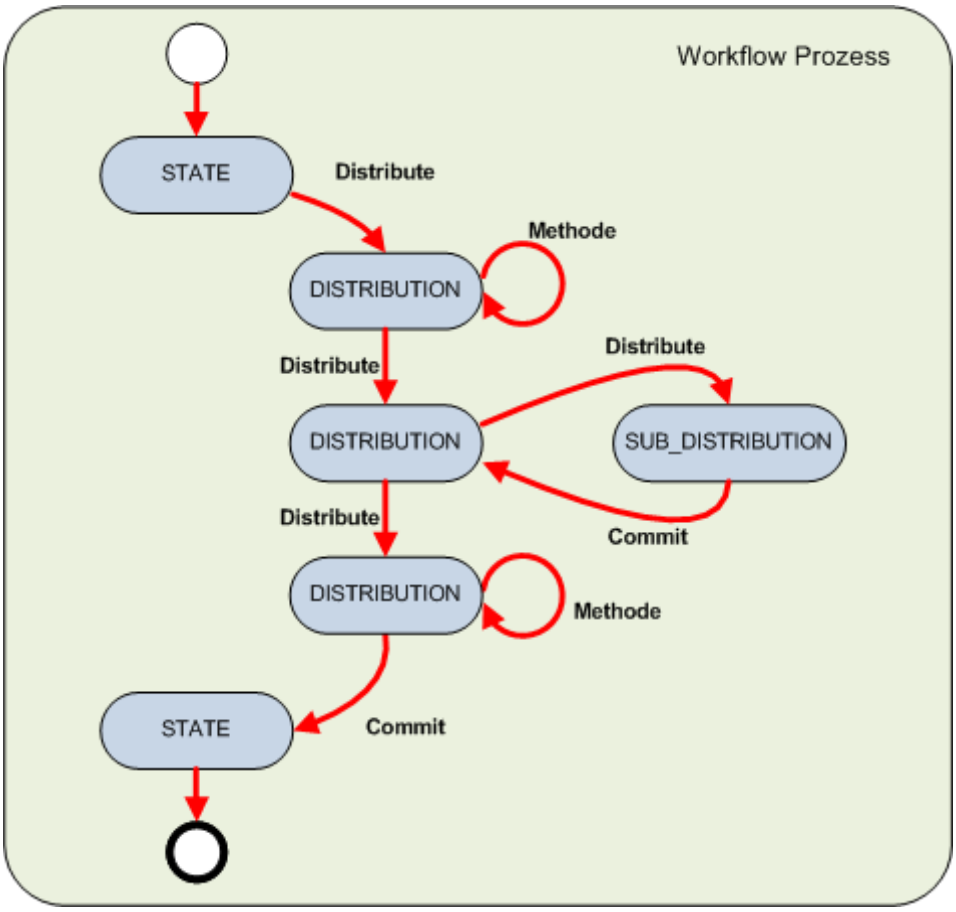


Feld	Bedeutung
ACTVITY_ID	Aktivität mit der Methode *Object.Update()* auf die sich der Eintrag bezieht.
SUB_METHODE_NAME	leer
OBJ_TYPE	Festlegen, auf welches beteiligte Objekt sich der Eintrag bezieht und auf das die Aktion angewendet werden soll. «this»: Container-Objekt, in welchem das Objekt abgelegt wird «clipboard»: aktuelles Objekt
Restliche Felder	s. Abschnitt «*Cut&Paste Operation > Prozesswechsel* (Seite 262)»

Distribution

Der Workflow-Prozess wird durch zwei Charakteristika bestimmt:

- Die Schritte von Prozesszustand zu Prozesszustand können nur durch die Methoden *Distribute()* und *Commit()* ausgelöst werden.
- Die Prozessstatus können nur vom Statustyp « DISTRIBUTION » und «SUB_ DISTRIBUTION » sein, ausser Start- und Endpunkt.



Die Workflow Methoden

Nur die Methoden *Distribute()* und *Commit()* bringen den Workflow in einen neuen Prozesszustand. Dies kann bedeuten, dass der Teilnehmerkreis ändert, den Teilnehmer neue Aufgaben zugewiesen werden müssen und Notifikationen verschickt werden müssen. Diese Funktionen werden mit diesen beiden Methoden implementiert.

Mit der Methode *Commit()* wird der Workflow-Prozess beendet.

Alle anderen Methoden können natürlich im Workflow-Prozess verwendet werden, führen aber nicht zu einer Änderung des Prozessstatus und damit zu keiner Änderung im Workflow.

Die Workflow Statustypen

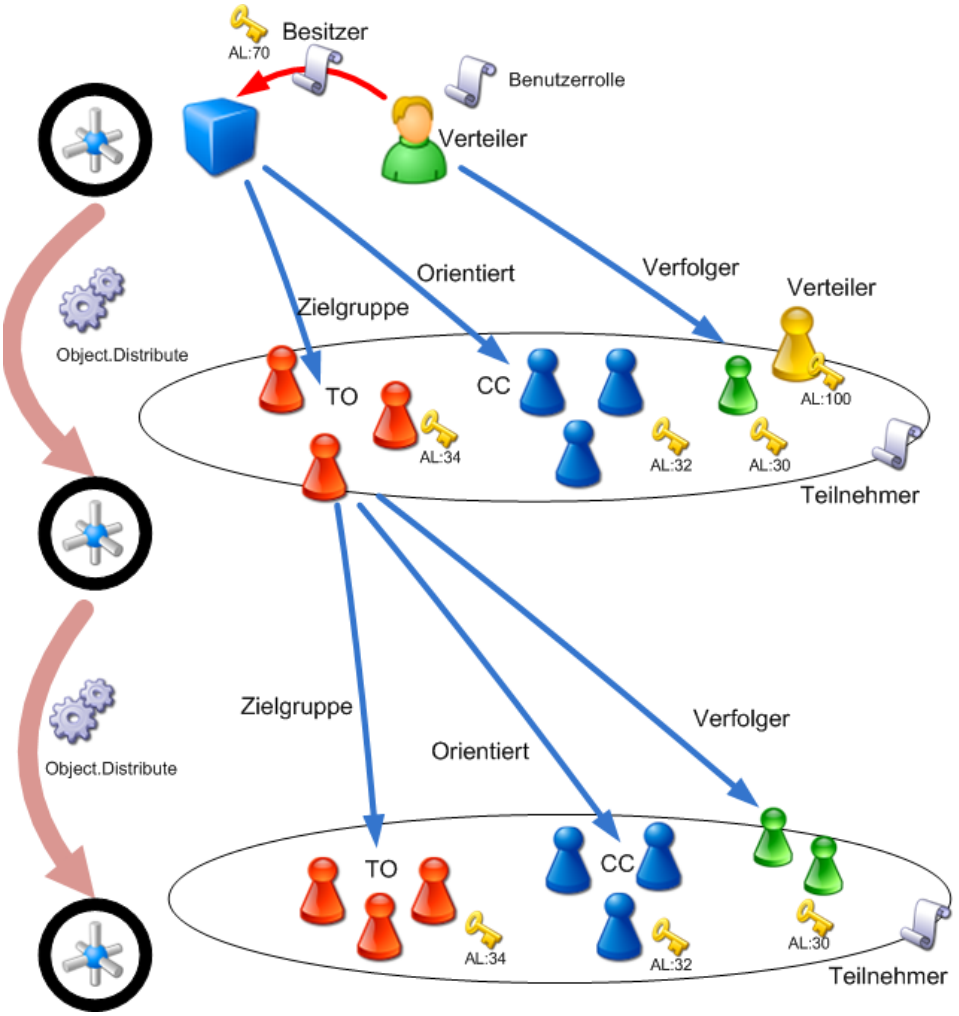
Jeder Prozess startet in einem Status des Typs «STATE». Ab diesem Punkt kann der Workflow-Prozess mit den Methoden *Distribute()* gestartet werden. Diese bringen den Prozess von Status zu Status, die den Typ «DISTRIBUTION» oder «SUB_DISTRIBUTION» aufweisen.

Für das Zusammenführen oder Aufsplitten von Prozesszweigen wird der Status mit einem «CONDITIONAL» Prozessstatus erweitert, das die Synchronisation der Zweige oder starten neuer Threads übernimmt.

Die Methode *Object.Commit()* bringt am Schluss den Workflow-Prozess vom Status des Typs «DISTRIBUTION» in einen des Typs «STATE». Diese Methode beendet auch einen Status «SUB_DISTRIBUTION» und der Workflow kehrt in den ursprünglichen Status «DISTRIBUTION» zurück.

Dynamische Systemrollen

Im Workflow-Prozess werden üblicherweise keine statischen Rollen eingesetzt. Durch die Parameter der Object.Distribute() Methode wird im Customizing festgelegt, welche Benutzer oder Rollen im nächsten Prozessstatus eine Funktion übernehmen.



Der Initiant – Distributor

Dieser Benutzer löst die Distribute-Methode aus. Der entsprechenden Aktivität ist die auch seine Benutzerrolle zugewiesen. Er kann auch der Besitzer des Objektes sein.

Zielgruppe – TO-Member

Der Initiant bestimmt eine oder mehrere Rollen. an welche der Prozess zur Weiterarbeit weitergereicht werden soll. Üblicherweise kann nur die Zielgruppe verändernde Aktivitäten am Objekt ausführen. Und nur Teilnehmer aus dieser Gruppe können Initiant für den nächsten Folgeschritt sein. Zielgruppen Teilnehmer erhalten autom. einen Access-Level von 34.

Orientierte Gruppe – CC-Member

Der Iniitiant kann zusätzlich eine oder mehrere Rollen festlegen. die im nächsten Prozessstatus orientiert werden. Üblicherweise kann diese Gruppe nur lesende Aktivitäten am Objekt ausführen und die Teilnehmer können den Workflow Prozess nicht beeinflussen. Diese Teilnehmer erhalten autom. einen Access-Level von 32.

Verfolger Gruppe

Der Initiant eines *Distribute* kann sich als Verfolger eintragen. In späteren Schritten können Statusänderungen an die Verfolger zurückgemeldet werden. Verfolger haben einen Access-Level von 30. Die aktuelle Verfolgergruppe bleibt im nächsten Prozessschritt erhalten, so dass die Rückmeldung an vorherige Initianten möglich bleibt. Die Weitergabe wird unterbrochen, wenn in einer Aktivität keine Distribute-Methode verwendet wird. Deshalb muss im gesamten Workflow Prozess die Methode *Object.Distribute()* verwendet werden.

Teilnehmer

Die Mitglieder dieser drei Gruppen sind alle in der dynamischen Systemrolle «Teilnehmer» enthalten. Damit den Teilnehmern eines Distribute die notwendigen Aktivitäten zur Verfügung stehen, wird den Aktivitäten am besten jeweils die Systemrolle «Teilnehmer» zugewiesen.

Aufsetzen der Parameter

Für die Verwendung der Methode *Object.Distribute()* müssen folgende Parameter aufgesetzt sein:

Parameter	Bedeutung
An	Zielgruppe: Rollen, welche im nächsten Prozessstatus einen Workflow Aktivität ausführen sollen. Es muss die Liste der betreffenden IDs bereitgestellt werden.
Cc	Orientierte Gruppe: Rollen, welche im nächsten Prozessstatus eine «Lesende» Aktivität ausführen sollen. Es muss die Liste der betreffenden IDs bereitgestellt werden.
Auftrag	Formulierter Text für Zielgruppe: Anleitung für die durchzuführende Aktion.
Ablaufdatum	Maximale Wartezeit: Datum bis die Aktivität zu erfolgen hat. Nach Ablauf wird ein Ereignis ON_EXPIRATION ausgelöst, welches die notwendige Behandlung durchführen kann.
Mitteilungsdatum	Fälligkeits Hinweis: Datum wann eine automatische Mitteilung erfolgt. Nach Ablauf wird ein Ereignis ON_NOTIFICATION ausgelöst, welches die notwendige Behandlung durchführen kann.

Parameter	Bedeutung
Verfolgung	Initiant in Verfolgergruppe: der Auslöser des *Distribute* wird in die Gruppe eingetragen.
Hinweise	Kurztexte für Aufgabenliste: Es können unterschiedliche Texte für Zielgruppe, Orientierte Gruppe, Verfolger und Initiant festgelegt werden. Diese werden im Protokoll und die in der Aufgabenliste verwendet.

Aufsetzen über Attributprofil

Die definieren der Parameter über das Attributprofil der Aktivität ist die einfachste Möglichkeit. Die einzelnen Werte können durch Abfragen beim Initianten, automatisch durch SQL-Statement oder fix in versteckten Feldern aufgesetzt werden.

In der Maske sind folgende Attribute notwendig:

Parameter	Attribut
An	temp.to_chosen -> ID Liste der Rollen
Cc	temp.cc_chosen -> ID Liste der Rollen
Auftrag	temp.task -> Textstring
Ablaufdatum	temp.next_expiration_date -> absolutes Datumsfeld
Mitteilungsdatum	temp.next_notification_date -> absolutes Datumsfeld
Verfolgung	temp.followup -> «YES» | «NO»
Hinweise	temp.summary_member_to -> Textstring temp.summary_member_cc -> Textstring temp.summary_member_fu -> Textstring temp.summary_member_distributor -> Textstring

Zudem muss im Show-Type BUTTON auch die «SENDEN» Schaltfläche richtig aufgesetzt sein:

- Tag Prefs = «method/object_distribute_files/object_distribute1.jsp»

Aufsetzen über Parameter-Statement

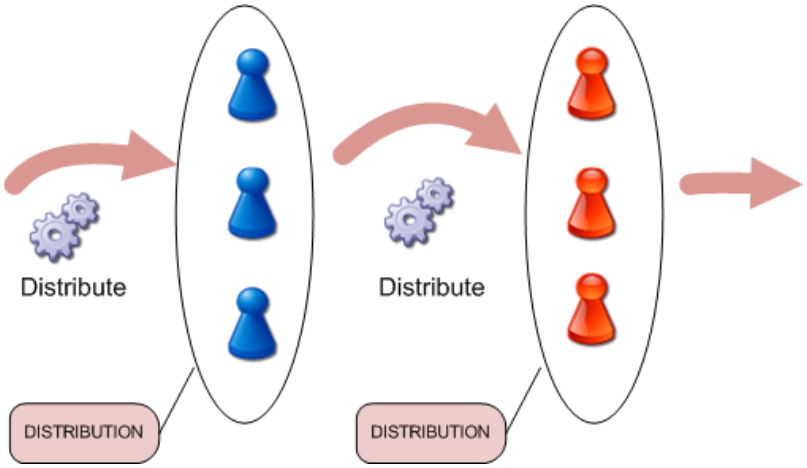
Wenn die Methode *Object-Distribute()* in einer Fork-Behandlung ausgelöst werden muss, können die notwendigen Parameter nicht über ein Attributprofil aufgesetzt werden. In diesem Fall muss ein PARAMETER-STMT SQL-Statement die erforderlichen Parameterwerte zurückliefern.

Weitere Informationen dazu im Abschnitt «*SQL-Statements* > *Parameter-Statement* (Seite 48)»

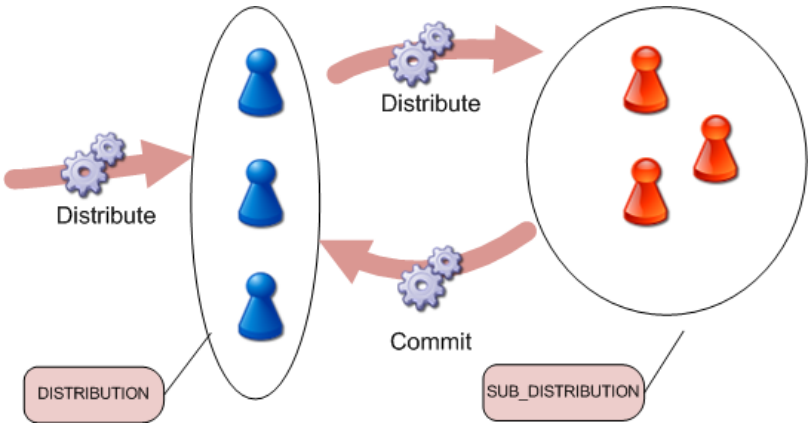
Unterschied Distribution – SubDistribution

Unterschied Distribution – SubDistribution

Bei einem *Distribute()* wird das Objekt an mehrere Teilnehmer verteilt. Wird von einem Teilnehmer ein erneuter *Distribute()* an eine andere Gruppe ausgelöst, wird bei all bisherigen Teilnehmern die Verteilung abgeschlossen. Bei einem «Distribute» ist der Typ des nächsten Prozesstatus auf «DISTRIBUTION» gesetzt.

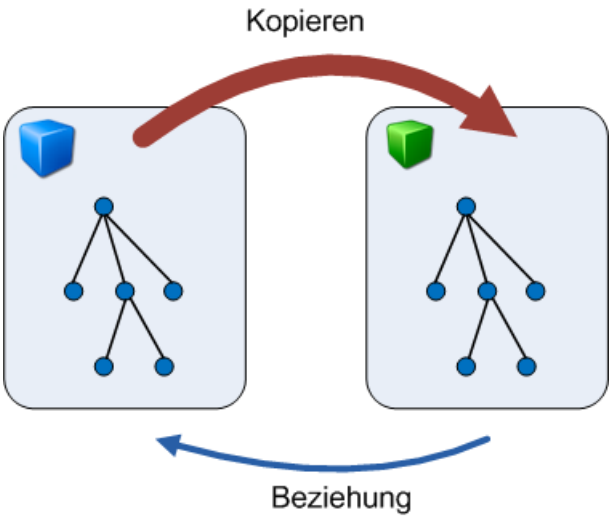


Bei einem «Subdistribute» ist der Typ des nächsten Prozessstatus auf «SUB_DISTRIBUTION» gesetzt. Mit *Distribute()* kann ein Teilnehmer ebenfalls eine erneute Verteilung an eine andere Teilnehmergruppe vornehmen. Bei den bisherigen Teilnehmern wird die Verteilung jedoch nicht abgeschlossen, sondern nur gesperrt. Hat die neue Teilnehmergruppe ihrerseits die Verteilung abgeschlossen, kann der ursprüngliche Teilnehmerkreis – nach einem *Commit()* - wieder daran weiterarbeiten.

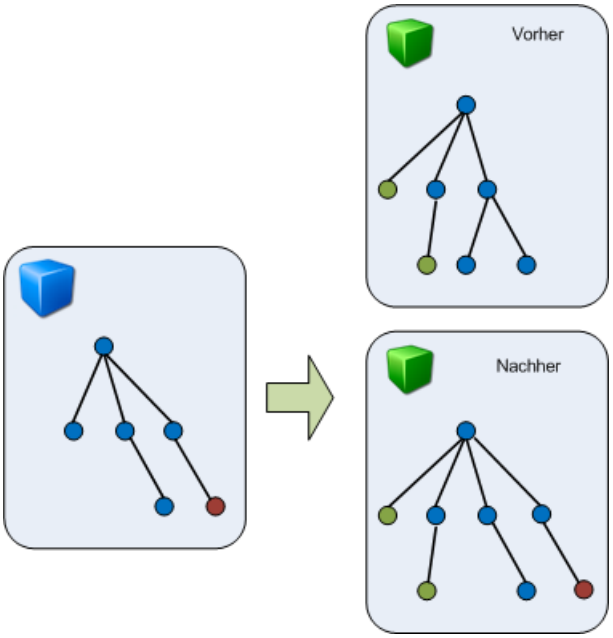


Clone Methoden

Die Methode *Tree.Clone()* erstellt von einem Objekt mit allen untergeordneten Objekten eine vollständige Kopie. Im Unterschied zur normalen Kopier-Methode, kennt das neue Objekt nach dessen Erzeugen weiterhin sein Ursprungsobjekt.



Dank dieser Beziehung kann zu einem späteren Zeitpunkt, eine Synchronisierung zwischen dem Ursprungsobjekt und der Kopie erfolgen. In der Praxis können mit diesem Mechanismus Änderungen an einer Originalvorlage automatisch an allen daraus abgeleiteten Strukturen nachgeführt werden. Beispielsweise kann die Struktur von Dossiers zentral definiert und in allen existierenden Dossiers nachvollzogen werden.



Mit der Methode *Clone.Synchronize()* werden die Strukturen von abgeleiteten Kopien entsprechend der Originalstruktur nachgeführt:

Original Objekt	Kopie Objekt
	Neues Objekt wurde erstellt: -> bleibt erhalten
	Objekt aus Original-Objekt gelöscht: -> wird wieder hergestellt
Neues Objekt	-> Objektstruktur wird ergänzt
Objekt gelöscht	Objekt enthält keine Objekte, die nach dem Kopieren erzeugt wurden: -> Objekt wird gelöscht Objekt enthält Objekte, die nach dem Kopieren erzeugt wurden: -> Objekt bleibt erhalten
Objekt in Struktur verschoben	Objekt wird in Struktur ebenfalls verschoben

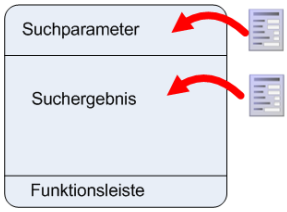
Hinweis

- Wenn *Copy.Clone()* verwendet wird um eine neue Vorlagenstruktur zu erzeugen, ist meistens keine Beziehung zum Originalobjekt gewünscht. In diesem Fall muss die Methode *Copy.Clone()* mit dem Modus «Einträge in Clone Tabelle löschen» ausgeführt werden.

Sucher

Zur Definition eines Suchers sind folgende 4 resp. 5 Schritte notwendig:

1. Festlegen eines Attributprofils für die Eingabe der Suchparameter.
2. Definition des SQL-Befehls, welcher die Suche durchführen soll.
3. Festlegen eines Attributprofils zur Anzeige des Suchergebnisses.
4. Sucher erzeugen
5. ggf. als Pop-up-Sucher in Formular integrieren.



1. Die Suchmaske definieren

Definieren Sie ein Attributprofil, welches die notwendigen Eingabefelder für die Suchmaske enthält.

Der Objekt-ID der neuen Suchmaske wird notiert. Dieser wird im Listeneintrag der Attributprofile angezeigt.

2. Den SQL Befehl für die "Suche"

In der Tabelle SQL_STMT wird ein neuer Eintrag für den SQL-Befehl vom Typ «FINDER» vorgenommen, welcher das Suchresultat liefern soll. Für die Erzeugen eines SQL-Statements s. Abschnitt «*Generelles Arbeiten* > *SQL-Statements* (Seite 33)»

Aufbau des SQL-Befehls:

Feldliste	fix vorgegeben: *object.obj_id,* *objclass.super_img_colapsed,* *objclass.objclass_name,* *ss.type,* *object.classification* *?attribList,*
Tabellen	relativ fix *sys_object, sys_attribute, sys_objclass,* *sys_structure ss,* ggf. weitere Tabellen
Where	join derTabellen

	sys_object.obj_id = ss.lower and *sys_object.obj_id = sys_attribute.obj_id and* *sys_attribute.page_nr = 0 and* *sys_object.objclass_id = sys_objclass.objclass_id and* *ss.type = 'ORIGINAL'* sowie eigentliches Suchkriterium

Hinweise

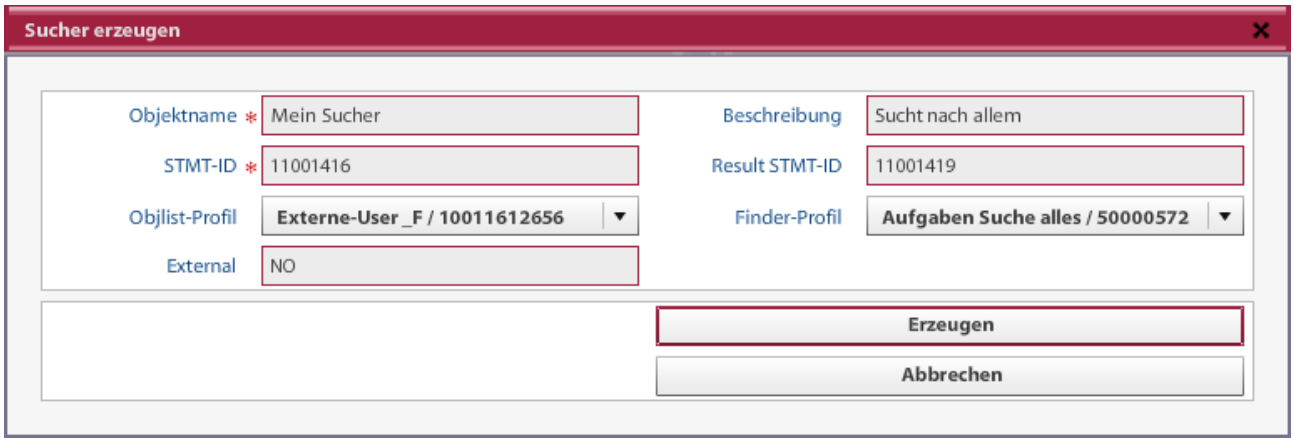
- Die Erstellung eines korrekten SQL-Befehls bedingt einen *i-engine* Experten.
- Der Parameter ?attribList wird vor Ausführung des SQL-Statements durch die Einträge im Attributprofil ersetzt. Dabei werden die Felder der BODY-Einträge in Reihenfolge der Sequenz in das SELECT Statement eingefügt. Maximal können 10 Felder angezeigt werden.

3. Die Ergebnismaske festlegen

Notieren Sie sich auch die Objekt-ID der Ergebnismaske.

4. Sucher erzeugen

Im Ordner für die Ablage von Sucher */Basismodule/Sucher* wird über das Kontextmenü **Sucher Erzeugen** die Eingabemaske aufgerufen:



Feld	Bedeutung
Objektname	Identifikation des Suchers
Beschreibung	Falls der *Objektname* nicht eindeutig ist oder die maximale Länge von 12 Zeichen überschreitet, wird ein innerhalb des Moduls eindeutiger Mnemonic in diesem Feld festgehalten.
STMT-ID	Das SQL-Statement vom Typ «FINDER» liefert die Such-

Feld	Bedeutung
	ergebnisse, die als Listenanzeige ausgegeben werden.
Result STMT-ID	Das SQL-Statement vom Type «RETURN_STMT» übergibt das ausgewählte Element an den Aufrufer zurück. Diese Eingabe wird benötigt, wenn der Sucher als Pop-up-Sucher eingesetzt wird. Wenn leer wird der Objekt-ID zurückgeliefert.
ObjList Profil	ID des Attributprofils das zur Anzeige der Sucherresultate verwendet wird.
Finder Profil	ID des Attributprofils das zur Abfrage der Suchparameter verwendet wird.
External	Sucher können durch externe Webanwendungen benutzt werden. Dieses Feld kann von der Webanwendung genutzt werden, um jene Sucher auszufiltern, die einsetzbar sind. «YES»: von externer Anwendung zu benützen «No»: nicht benützen

5. Den neuen Sucher integrieren

Der neue Sucher steht sofort im Container-Objekt zur Verfügung. Er kann auch als eigenständiges Eingabeelement über eine Schalfläche in einer Eingabemaske aufgerufen werden. Die Verwendung als Pop-up Eingabeelement bedingt zusätzliche Einstellungen:

- Es ist ein zusätzliches SQL-Statement vom Typ «RETURN_STMT» notwendig, welches aus dem/den gewählten Objekt-ID(s) die entsprechenden Objekt-ID / Anzeigetext Paare erstellt.
- Für das Maskenfeld müssen in der Tabelle ATTRIBUTE_SHOW folgende Felder aufgesetzt werden:
  POPUP=«POPUP»
  TAG=«FINDER»
  POPUP_OBJ_ID=Objekt-ID des Suchers

Eine weitere Möglichkeit ist die Anzeige im linken Frame; aufgerufen über eine Menü-Schalfläche. Im rechten Frame werden dann die Suchergebnisse aufgelistet.

- In der Tabelle PROPERTIES muss ein Menüeintrag erstellt werden, welche die Anzeige des Suchers in der Strukturanzeige auslöst. Details sind im Abschnitt «Menüleiste & Strukturanzeige» zu finden.
- Im Eintrag für KEY=«MENU_OBJ_ID» wird die ID des Ordners eingetragen, welcher den Sucher enthält.

- Im Eintrag für KEY=«MENU_URL» wird der Aufruf für den Sucher hinterlegt: /method/mrb.jsp?obj_id=<ID des Suchers>&activity_id<Methode object.find()>&parent_id=0

Standard-Sucher

- Im Win Client und Flex Client wird auf dem Schreibtisch ein Volltextsucher (1009012) angeboten.
- Dieser Sucher kann nicht angepasst werden.

Volltextsucher

Der Volltextsucher ermöglicht die Suche nach Objekten, welche im Inhalt oder in den Eigenschaftswerten den gesuchten Text enthalten. In der Objektklasse wird durch das setzen der Eigenschaft *Attrib.Indexierung* festgelegt, dass die zugehörigen Dokumente durch den Volltextsucher durchsucht werden. s. «*Eine Objektklasse erzeugen* (Seite 164)»

Bei der Eingabe der Suchbegriffe werden folgende Definitionen berücksichtigt:

«*»	Jokerzeichen für eine beliebige Anzahl Zeichen Das Jokerzeichen kann nicht am Wortanfang stehen
«#»	Jokerzeichen für genau 1 Zeichen, kann mehrfach eingefügt werden. Das Jokerzeichen kann nicht am Wortanfang stehen
«AND» «OR» «NOT»	Operatoren für das Verknüpfen von Suchbegriffen. Beim Auflösen gilt die Reihenfolge NOT > AND > OR. Fehlt ein Operator zwischen den Suchbegriffen, wird ein AND angenommen.
«(» «)»	Mit runden Klammern können die Suchbegriffe gruppiert und so die Auswertungsreihenfolge klar gemacht werden.
«"»	Mit Gänsefüsschen wird explizite ein Suchbegriff definiert, der sich aus mehreren Wörtern zusammensetzt.

Weiter gilt:

- Gross/Kleinschreibung wird nicht berücksichtigt
- Ohne Jokerzeichen werden nur exakte Suchbegriffe gefunden.
- Die Reihenfolge der Suchbegriffe ist egal.
- Nach den Zeichen «-» und «_» kann nicht gesucht werden - auch nicht mit Begriffen in Gänsefüsschen. D.h. diese Zeichen werden durch einen Leerschlag ersetzt, was zu einer AND-Suche mit einzelnen Begriffen führt.
- Werden mehrere Suchbegriffe verwendet, müssen entweder alle Begriffe in den Eigenschaftswerten ODER alle Begriffe im Inhalt vorhanden sein.

Constraint - Filter

Im Unterschied zu einem Sucher sind die Suchkriterien bei einem Constraint vom Benutzer nicht beeinflussbar. Es wird keine Eingabemaske für die Suchparameter angezeigt, sondern direkt das Suchergebnis präsentiert. Die notwendigen Aufgaben vereinfachen sich deshalb etwas.

Zur Definition eines Constraints sind folgende 3 Schritte notwendig:

1. Definition des SQL-Befehls, welcher die Suche durchführen soll.
2. Festlegen eines Attributprofils zur Anzeige des Suchergebnisses.
3. Constraint erzeugen

1. Den SQL Befehl für den "Filter"

In der Tabelle SQL_STMT wird ein neuer Eintrag für den SQL-Befehl vom Typ «CONSTRAINT» vorgenommen, welcher das Suchresultat liefern soll. Für die Erzeugen eines SQL-Statements s. Abschnitt «*Generelles Arbeiten* > *SQL-Statements* (Seite 33)»

Der Aufbau des SQL-Statements entspricht jenem für einen Sucher.

2. Die Ergebnismaske festlegen

Definieren Sie das Attributprofil für die Anzeige der Ergebnisliste.

3. Constraint erzeugen

Im Ordner für die Ablage von Sucher wird über das Kontextmenü **Constraint Erzeugen** oder **Sucher Erzeugen** die Eingabemaske aufgerufen:

Feld	Bedeutung
Objektname	Identifikation des Constraint
Beschreibung	Falls der *Objektname* nicht eindeutig ist oder die maximale Länge von 12 Zeichen überschreitet, wird ein innerhalb des Moduls eindeutiger Mnemonic in diesem Feld festgehalten.
STMT-ID	Das SQL-Statement vom Typ «CONSTRAINT» liefert die Suchergebnisse, die als Listenanzeige ausgegeben werden. Der Aufbau des SQL-Befehls entspricht jenem des Suchers.
Result STMT-ID	Nicht verwendet.

Feld	Bedeutung
ObjList Profil	ID des Attributprofils das zur Anzeige der Sucherresultate verwendet wird.
Finder Profil	Nicht verwendet
External	Nicht verwendet

Setzen Sie noch die Sichtbarkeit des neuen Objektes auf.

Falls der Constraint über die Funktion **Sucher erzeugen** erstellt wurde, sind noch folgende Korrekturen notwendig:

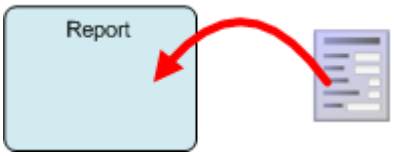
- Die Objektklasse des neuen Objektes muss auf eine Objektklasse des Typ «Constraint» gesetzt werden. Standardmässig wird Objektklasse «Bedingung» verwendet.
- Der Sucher Eintrag in der Tabelle FINDER muss vom Typ «FINDER» auf «CONSTRAINT» gewechselt werden.

Hinweis

- Bei Sucher und Constraints werden in der Ergebnisliste nur jene Einträge angezeigt, auf die der Benutzer Zugriffsrechte hat.
- Ein neuer Constraint wird oft durch das Kopieren eines vorhandenen erzeugt. Dies hat den Vorteil, dass z.B. Attributprofil und Berechtigungen bereits aufgesetzt sind.
- Der Constraint kann nur eingesetzt werden, wenn die resultierende Anzahl Einträge unter 1000 bleibt.

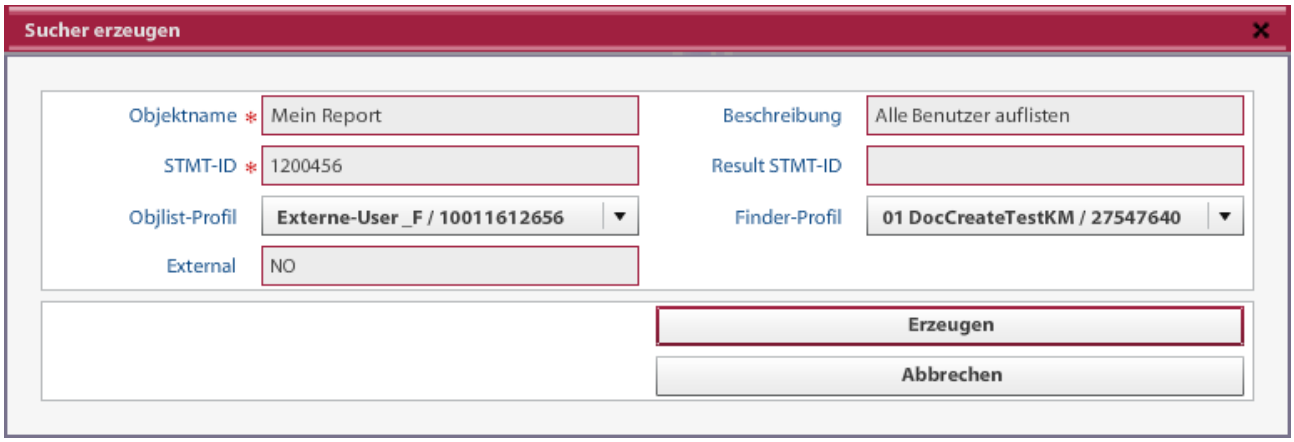
Reports

Reports erweitern die Darstellungsmöglichkeiten von Listen wesentlich. Im Attributprofil können über Zusatzangaben Gruppierungen und Sortierungen festgelegt werden. Die Datenabfrage wird über ein SQL-Statement des Typs «REPORT» ausgeführt. Der Aufbau des SQL-Befehls entspricht jenem für Sucher.



Neuen Report anlegen

Abhängig vom Scope wird der Report in der Struktur im jeweiligen «Sucher» Verzeichnis erstellt. Über das Kontextmenü **Bericht erzeugen** wird die Eingabemaske angezeigt.



Feld	Wert
Objektname	Identifikation des Reports
Beschreibung	Falls der *Objektname* nicht eindeutig ist oder die maximale Länge von 12 Zeichen überschreitet, wird ein innerhalb des Moduls eindeutiger Mnemonic in diesem Feld festgehalten.
STMT-ID	Das SQL-Statement vom Typ «REPORT» liefert die Suchergebnisse, die als Listenanzeige ausgegeben werden. Der Auf-

Feld	Wert
	bau des SQL-Befehls entspricht jenem des Suchers.
Result STMT-ID	Nicht verwendet.
ObjList-Profil	ID des Attributprofils das zur Anzeige der Sucherresultate verwendet wird.
Finder-Profil	ID des Attributprofils das zur Anzeige der Reportparameter verwendet wird.
External	Nicht verwendet

Die Berechtigung für das neue Objekt muss anschliessend noch korrekt aufgesetzt werden.

Attributprofile erzeugen

Im ObjList Attributprofil werden die Felder für die anzuzeigenden Kolonnen festgelegt. Dabei ist speziell zu beachten:

Feld	Bemerkung
SHOW_TYPE	muss für alle Einträge «BODY» sein
SEQ	Einträge lückenlos durchnummeriert sein, beginnend mit «1».
H_POS	Auch diese Einträge müssen lückenlos durchnummeriert sein.
V_POS	Muss fix auf «1» gesetzt werden
DISPLAYNAME	Kolonnentitel
Andere	Werden nicht verwendet

- Im Finder Attributprofil werden die Filterparameter für die Datenselektion des Reports abgefragt.
- Falls keine Parameterabfrage notwendig wird, darf das FinderProfil keine BODY Einträge enthalten. Damit wird direkt der Report angezeigt. In diesem Fall muss die Sortierung der Einträge durch das SQL-Statement vorgegeben sein.

Das SQL-Statement vom Typ «REPORT»

Das SQL-Statement muss genau die Anzahl Werte in der richtigen Reihenfolge zurückliefern, entsprechend den Kolonnen im Attributprofil. Jeder String für einen Kolonnenwert kann beliebige HTML-Syntax enthalten. Dh. es können auch Bilder als Kolonnenwerte oder Textauszeichnungen verwendet werden.

Liste strukturieren

Die Liste kann gruppiert und / oder sortiert werden. Diese Einstellungen werden ebenfalls im Attributprofil konfiguriert.

Gruppierung:

Die Einträge können nach Feldern gruppiert dargestellt werden. Für die Gruppierung stehen die im Attributprofil definierten Kolonnen zur Verfügung. Die Reihenfolge der Gruppen kann auf- oder absteigend erfolgen.

Feld	Bemerkung
SHOW_TYPE	«BODY»
ATTRIBUTE_NAME	«temp.groupby1» … <temp.groupby10» Maximal ist eine 10-stufige Gruppierung unterstützt. Der Wert entspricht der Kolonnennummer im Attributprofil
	«temp.groupbydir1»…«temp.groupbydir10» Wert >=0: aufsteigende Sortierung Wert <0: absteigende Sortierung

Sortierung:

Die Einträge innerhalb einer Gruppe können sortiert angezeigt werden. Es können bis zu 10 Sortierfelder angegeben werden.

Feld	Bemerkung
SHOW_TYPE	«BODY»
ATTRIBUTE_NAME	«temp.orderby1» … <temp.orderby10» Der Wert muss einer Kolonnennummer im Attributprofil entsprechen.
	«temp.sortbydir1»…«temp.sortbydir10» Wert >=0: aufsteigende Sortierung Wert <0: absteigende Sortierung

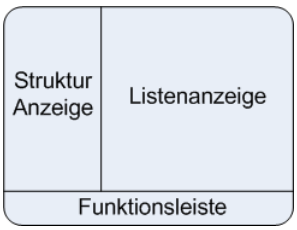
Die Gruppierungs- und Sortierungseinträge im Attributprofil werden nur ausgewertet, wenn das SQL-Statement mit dem Platzhalter «?orderby» ergänzt wird.

Hinweis

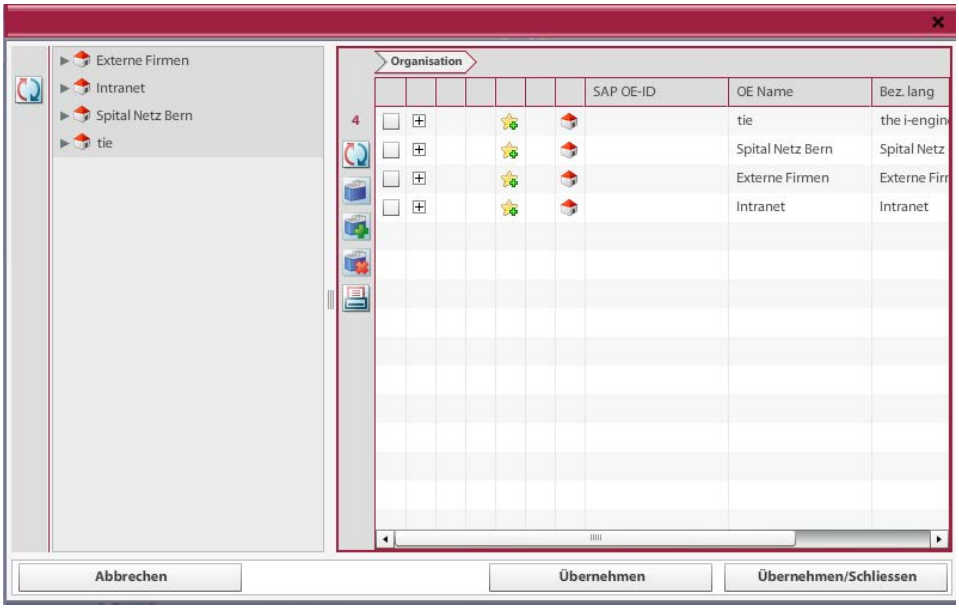
- Für die Ergebnislisten werden die Zugriffsrechte des Benutzers nicht automatisch berücksichtigt. Dh. das eingesetzte SQL-Statement muss diese Rechte ebenfalls auswerten.

- Reports können nur in einem eigenen Fenster angezeigt werden.
- Der Report kann nicht auf einem bestimmten Objekt ausgelöst werden. Um den Report für ein bestimmtes Objekt auszulösen, muss auf diesem Objekt die Methode *object.execute()* ausgelöst werden. Das SQL-Statement des Reports kann dann die Variable *?caller-id* - welche dem ursprünglichen Objekt entspricht - auswerten.

Pop-up Fenster definieren



In den Attributprofilen können separate Pop-up Fenster für die komfortable Auswahl verwendet werden. Das Pop-up Fenster zeigt im linken Frame die Strukturanzeige ab einem vorbestimmten Wurzelobjekt. Im rechten Frame werden die Objekte aufgelistet, wobei nur die wählbaren Einträge ein Auswahlkästchen aufweisen. Die Schalflächen der Funktionsleiste erlauben die direkte Übernahme der ausgewählten Werte in das Feld der ursprünglichen Eingabemaske.



DB Action

Für jedes Pop-up Fenster muss der Feldeintrag in der Tabelle ATTRIBUTE_SHOW ergänzt werden.

TAG	Fix: «POPUP»
POPUP_ OBJ_ID	Wurzelelement der Baumanzeige Der Strukturbaum wird ab dem gewählten Objekt angezeigt. Das Wurzelelement kann statisch als Objekt-ID vorgegeben oder dynamisch über ein SQL-Statement bestimmt werden. Je nachdem ist ein Objekt-ID oder ein SQL-Statement im Feld anzugeben. Über ein SQL-Statement kann das Wurzelelement z.B. abhängig vom aktuellen Benutzer ausgesetzt werden. Das SQL-Statement darf nur gerade eine einzige Objekt-ID zurückliefern.
TAG_PREF	Parameter für das Pop-up Fenster

	Fix: «menu=yes»
	Fix: «checkbox=yes» Anzeige der Auswahlkästchen für die Auswahl eines Eintrages

	«checkboxstatementid=…»
	
	Die Listenanzeige kann Objekte enthalten, welche nicht ausgewählt werden dürfen (z.B. falscher Objekttyp). Über ein SQL-Statement (SQL-Typ «Checkbox-Statement») wird eine List der Objekt-IDs erzeugt, die auswählbar sind. Nur Objekt-IDs die in dieser Liste sind, werden mit einem Auswahlkästchen angezeigt.
	
	Beispiel SQL-Statement:
	`SELECT ObjId FROM rolle`
	`WHERE roll_id IN (?Objlist)`
	
	Dem Parameter muss die ID des SQL-Statements mitgegeben werden.

	«multiselect=no» resp. «multiselect=yes» Unterscheidung ob nur ein Einzelwert oder auch mehrer ausgewählt werden können.
	Fix: Für Rückwärtskompatibilität «leafsonly=no» «noleafs=no»

VALUE_STMT_ID	Rückgabewert eruieren
	
	Oft ist das ausgewählte Objekt nicht direkt der benötigte Rückgabewert für das Eingabefeld. Deshalb wird via SQL-Statement (SQL_Typ «Value-Statement») aus dem ausgewählten Objekt der Rückgabewert bestimmt. Das SQL-Statement muss Speicherwert und Anzeigewert zurückliefern.
	
	`SELECT ObjID, ObjName FROM rolle`
	`WHERE objid IN (?obj_iD)`

POPUP_P1	Methode für Baumanzeige Für die Anzeige der Baumstruktur stehen 2 Methoden zur Verfügung: «10010209»: Object.Tree.ObjList «10010218»: Object.Tree2.ObjList Für die Beschreibung der Unterschiede s. Abschnitt «*Menüleiste & Strukturanzeige* (Seite 149)».

POPUP_ DIALOGPREFS	Window Parameter String Aufsetzen des Fensterlayout, Grösse etc.
POPUP_ PROFILE_ID	Attributprofil für Listenanzeige Die Darstellung der Objektliste im rechten Frame wird durch ein Attributprofil bestimmt. In diesem Feld ist die Attributprofil-ID einzugeben.

Resultatfeld über Auswahlliste füllen

Oft muss in einer Maske in einem Resultatfeld ein oder mehrere Objekte gesammelt werden. Falls nur gerade ein einziges Element selektiert werden soll, kann die Auswahl über die Dropdown-Liste erfolgen – in allen anderen Fällen muss die Auswahl in separaten Suchfeldern erfolgen. Aus Komfortgründen werden dem Benutzer die Auswahlmöglichkeiten häufig in einem – oder auch mehreren – Auswahllisten zur Verfügung gestellt. Je nach Methode und Anwendungsanforderungen stehen unterschiedliche Mechanismen für die Implementierung zur Verfügung.

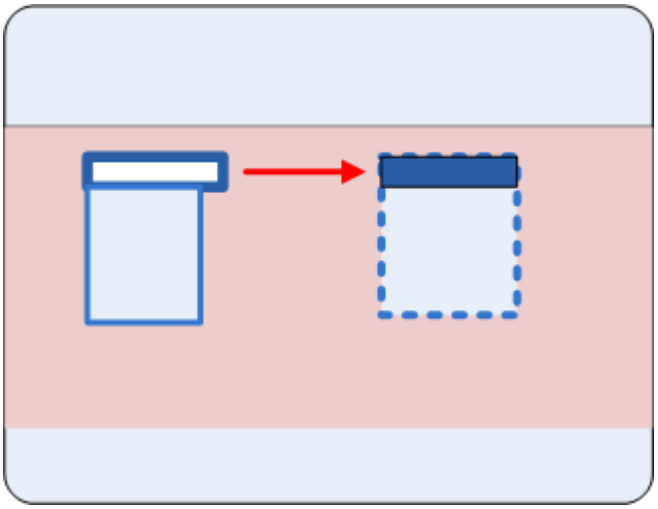
In den folgenden Beispielen sind Suchfeld und Resultatfeld immer separat dargestellt. Prinzipiell können das Such- und Resultatfeld aber in einem einzigen Attributfeld kombiniert werden. Vorausgesetz:

- das Resultatfeld enthält nur einen einzigen Wert
- das Resultatfeld wird nur von einem einzigen Resultatfeld befüllt.

Ein getrenntes Suchfeld ist beim Öffnen immer leer. In einem Resultatfeld – und kombinierten Feldern – werden aber die bisherigen Auswahlen angezeigt. Die initiale Anzeige muss entsprechend vorbereitet werden.

Dropdown Liste

Ein Eingabeelement mit *Tag* SELECT unterstützt nur die Auswahl eines einzigen Elementes. Für eine Mehrfachauswahl müssen die ausgewählten Elemente jeweils in einem Resultatfeld gesammelt werden.



Suchfeld

- *Tag* ist auf «SELECT» gesetzt -> Dropdown Liste
- *Reference* Wert für Suchfeld und Resultatfeld identisch.

- Anzeige: In *Popup_ObjID* ist die ID für ein SQL-Statement definiert, welches die Dropdown-Liste mit Werten füllt. Das SQL-Statement ist vom Type POPUP. Es liefert ObjID und als zweiten Wert den String für die Anzeige der Werte in der Auswahlliste.

Resultatfeld

- *Tag* ist auf «MULTICHOSEN» oder »SINGLECHOSEN» gesetzt
- *Reference* Wert für Suchfeld und Resultatfeld identisch.
- *Mandatory* auf «YES» (meistens)
- *Tag Prefs* auf « disabled » gesetzt.
  zusätzlich kann dort mit «minCard=n maxCard=n» die minimal notwendige resp. maximale Anzahl Einträge festgelegt werden.
- Anzeige: Für die Anzeige in der Liste werden direkt die Anzeigewerte aus dem Suchfeld übernommen. Dh. entsprechend dem Rückgabewert des SQL-Statements im Suchfeld.
- Initiale Anzeige: Ein SQL- Statement – vom Typ VALUE – liefert die initiale Liste der Werte für das Resultatfeld. Der ID des SQL-Statements wird im Feld *Value* des Resultatfeldes eingetragen.
  Bei einem Resultatfeld mit Mehrfachselektion wird die Liste der Objekt-IDs in einem «text» oder «long» Attributfeld abgespeichert. Der Parameter {?list} liefert die Liste der IDs.

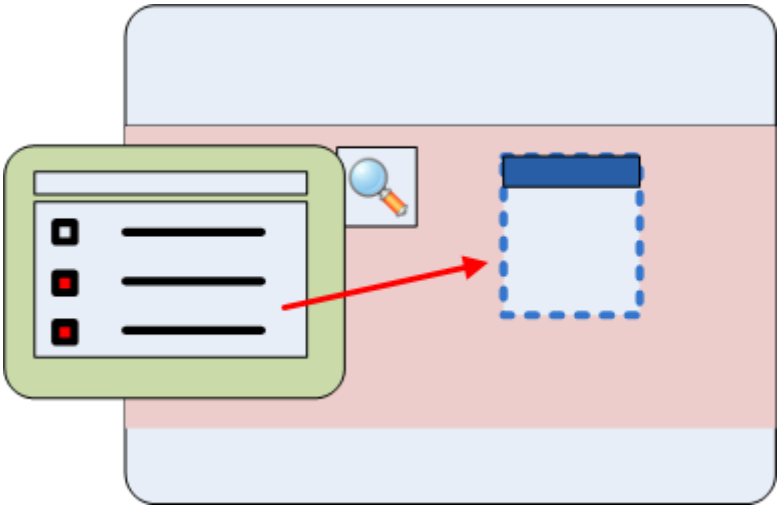
  Value SQL-Statement:

```
select a || ' ' || b || ' '+|| c from sys_attribute whe-
re id in (?list)
```

  Die Möglichkeit über «INTERNAL_DISPLAY_STATEMENT_ID» in der Tabelle OBJECT_PROPERY das SQL-Statement aufzusetzen, sollte nicht mehr verwendet werden.

Sucher oder Constraint

Eine komfortable Auswahl ist über einen separaten Sucher oder Constraint möglich. Die gewünschten Einträge werden in der Liste markiert und dann in das Resultatfeld der ursprünglichen Maske übernommen.



Suchfeld (Schaltfläche)

- *Tag* ist auf «FINDER» gesetzt -> Sucher oder Constraint
- *Reference* Wert für Suchfeld und Resultatfeld identisch.
- In *Popup-ObjID* ist die ID des gewünschten Pop-up Objektes, welches die Auswahlliste anzeigt, eingetragen.

Sucherfenster

- Im Pop-up Fenster ist ein entsprechende Element – ObjList – angezeigt, welches über Auswahlboxen die Markierung der gewünschten Einträge ermöglicht. Ein SQL-Statement vom Typ «FINDER» liefert die Anzeigeliste (ObjID + Anzeigestring) für die Auswahl. Die Anzeigeliste wird automatisch mit den notwendigen Auswahlboxen ergänzt.
- Ein SQL-Statement vom Typ «RETURN_STMT» liefert die markierten Werte zurück (ObjID + Anzeigestring).

Resultatfeld

- *Tag* ist auf «MULTICHOSEN» oder »SINGLECHOSEN» gesetzt
- *Reference* Wert für Suchfeld und Resultatfeld identisch.
- *Mandatory* auf «YES» (meistens)
- *Tag Prefs* auf « disabled » gesetzt.
  zusätzlich kann dort mit «minCard=n maxCard=n» die minimal notwendige resp. maximale Anzahl Einträge festgelegt werden. Über «size=x» wird die Anzahl Anzeigezeilen definiert.
- Anzeige: für die Anzeige in der Liste werden die Ausgaben aus dem «RETURN_STMT» des Suchfensters übernommen
- Initiale Anzeige: Ein SQL- Statement – vom Typ VALUE – liefert die initiale Liste der Werte für das Resultatfeld. Der ID des SQL-Statements wird im Feld *Value* des Resultatfeldes eingetragen.

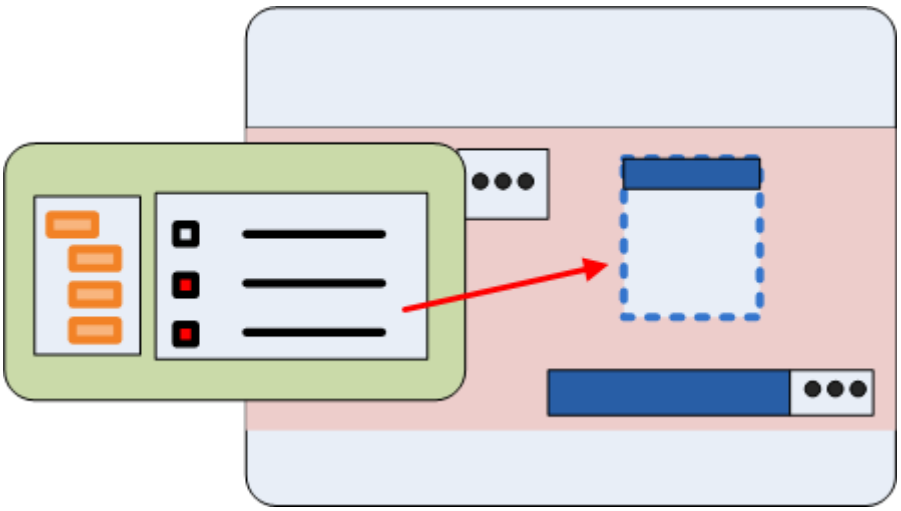
Bei einem Resultatfeld mit Mehrfachselektion wird die Liste der Objekt-IDs in einem «text» oder «long» Attributfeld abgespeichert. Der Parameter «?list» liefert die Liste der IDs.

Die Möglichkeit über «INTERNAL_DISPLAY_STATEMENT_ID» in der Tabelle OBJECT_PROPERY das SQL-Statement aufzusetzen, sollte nicht mehr verwendet werden.

Pop-up Fenster

In einem separaten Pop-up Fenster werden die möglichen Werte in einer Liste angezeigt und können entsprechend markiert werden. Das Wurzelobjekt für die Anzeige des Auswahlbaumes wird beim Customizing festgelegt.

In diesem Anwendungsfall wird oft das Such- und Resultatfeld zusammengefasst.



Suchfeld (Schaltfläche)

- *Tag* ist auf «POPUP» gesetzt -> Pop-up Fenster
- *Reference* Wert für Suchfeld und Resultatfeld identisch.
- Anzeige für das Resultatfeld: Wie die ausgewählten Werte im Resultatfeld angezeigt werden, wird in *Popup-ObjID* des Suchfeldes festgelegt. Das betreffende SQL-Statement ist vom Type VALUE_STMT. Es liefert ObjID und als zweiten Wert den String für die Anzeige.
  Das SQL-Statement wird eine Konstruktion wie
  `where id in (?objid)`
  verwenden, um nur die markierten Einträge auszuwählen. Der Parameter «?objid» enthält die vom Pop-up zurückgemeldeten Object-IDs. Da es sich um eine Liste handelt, wird er im SQL-Befehl in Klammern gesetzt: «(?objid)».

Pop-up Fenster

- Im Pop-up Fenster werden in der Strukturanzeige die Objekte ab dem vorgegebenen Wurzelelement angezeigt. Über Auswahlboxen können die ge-

wünschten Einträge ausgewählt werden. Ein SQL-Statement vom Typ «FINDER» liefert die Werteliste (ObjID + Anzeigestring) für die Auswahl.

- In der Tabelle ATTRIBUTE_SHOW muss u.a. im Feld POPUP_OBJ_ID der Objekt-ID des Wurzelelementes aufgesetzt werden. Zudem sind weitere Einstellungen z.B. im TAG_PREF Feld notwendig. s. dazu Abschnitt «*Pop-up Fenster definieren* (Seite 288)».

Resultatfeld

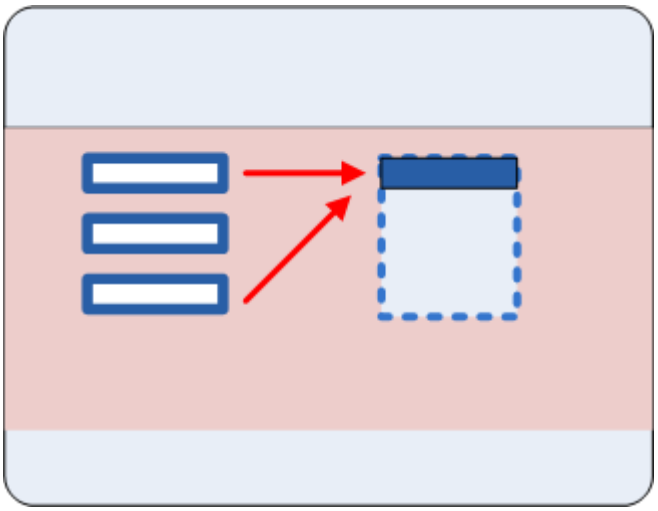
- *Tag* ist auf «MULTICHOSEN» oder »SINGLECHOSEN» gesetzt
- *Reference* Wert für Suchfeld und Resultatfeld identisch.
- *Mandatory* auf «YES» (meistens)
- *Tag Prefs* auf «disabled» gesetzt. (wenn Such- und Resultatfeld getrennt) zusätzlich kann dort mit «minCard=n maxCard=n» die minimal notwendige resp. maximale Anzahl Einträge festgelegt werden.
- Anzeige: wird im Suchfeld festgelegt
- Initiale Anzeige: Ein SQL- Statement – vom Typ VALUE – liefert die initiale Liste der Werte für das Resultatfeld. Der ID des SQL-Statements wird im Feld *Value* des Resultatfeldes eingetragen.
  Bei einem Resultatfeld mit Mehrfachselektion wird die Liste der Objekt-IDs in einem «text» oder «long» Attributfeld abgespeichert. Der Parameter {?list} liefert die Liste der IDs.

  Die Möglichkeit über «INTERNAL_DISPLAY_STATEMENT_ID» in der Tabelle OBJECT_PROPERY das SQL-Statement aufzusetzen, sollte nicht mehr verwendet werden.

Verwenden von CHOOSE-CHOSEN Show-Typen

Dieses Verfahren muss für einige Methoden verwendet werden: *Role.AddUser(), User.AddRole(), Object.ChangeOwner(), Object.Classify(), Object.SetAccess()* und *Object.SetShow().* Die aktuelle Implementation dieser Methoden bedingt, dass auf der Maske keine Register verwendet werden können. dh. nur der Show-Type BODY darf im Attributprofil verwendet werden.

Anwendungsbeispiel: Aus mehreren Suchfeldern – welche z.B. Auswahllisten nach unterschiedlichen Kriterien präsentieren – kann der Benutzer die gewünschten Einträge auswählen. Sie werden in einer Liste (resp. einem Feld, wenn nur ein einziges Element) gesammelt und durch die Methode abgespeichert.



Suchfeld

- *Show-Type* auf «CHOSE» gesetzt
- *Tag* auf «SELECT» gesetzt -> Dropdown-Liste
  Tag ist auf «FINDER» gesetzt -> Sucher oder Constraint
  Tag ist auf «POPUP» gesetzt -> Pop-up Fenster
- Anzeige: s. vorherige Ausführungen.

Resultatfeld

- *Show-Type* auf «CHOSEN» gesetzt
- *Tag* ist auf «MULTICHOSEN» oder »SINGLECHOSEN» gesetzt
  Die Methode *Object.ChangeOwner()* unterstützt keine Mehrfachauswahl.
- Anzeige: wird direkt aus dem Suchfeld übernommen.
- Initiale Anzeige: In der Tabelle OBJECT_PROPERY kann ein Eintrag für «INTERNAL_DISPLAY_STATEMENT_ID» vorgenommen werden. Das betreffende SQL- Statement – vom Typ VALUE – liefert dann die initiale Liste der Werte im Resultatfeld.

  Das Verfahren wird nur für die aufgeführten Methoden eingesetzt. Bei diesen kommt immer das gleiche SQL-Statement – 10001147 – zur Anwendung. Zudem wird die Anzeige mit weiteren Angaben durch die Methode selbst ergänzt.

Kollektoren

DB Action

In Kollektorelementen lassen sich unabhängige, logisch zusammenhängende Objekte verknüpfen. Der Benutzer wird damit ein direkter Zugriff auf alle erforderlichen Elemente, die er für seine Aufgabe benötigt, geboten. Kollektoren werden primär in Pop-up's zur Auswahl via Baumstruktur eingesetzt.

Kollektor erzeugen

Kollektoren sind immer kundenspezifische Erweiterung und werden darum im Bereich «Kunde > Parametrisierung > Kollektoren» abgelegt. Über das Kontextmenü **Kollektor erzeugen** wird ein neuer Eintrag erzeugt.

Das neue Kollektoren-Objekt ist von der Objektklasse «Kollektor»

Kollektor Einträge

Die benötigen Objekte werden über Cut&Paste von Links im Kollektor-Objekt angelegt.

Falls die Funktion **Verknüpfung erzeugen** in einem benötigten Objekt nicht implementiert ist, muss die Verknüpfung direkt in der Tabelle STRUCTURE eingetragen werden.

Feld	Eintrag
TYPE	Fix «LINK»
UPPER	Objekt-ID des Kollektors
LOWER	Objekt-ID des benötigen Objektes
TREE	«YES»
TREE2	«YES»
TREE3	«YES» - speziell wichtig, da ja der Kollektor in einem Popup verwendet wird.

Kollektor verwenden

Der Kollektor wird im Attributprofil verwendet. Die Verwendung erfolgt wie ein normales Popup-Fenster. s. dazu Abschnitt «*Popup Fenster definieren* (Seite 288)»

Verwendungsnachweis

Mit dem Verwendungsnachweis kann eine Liste abgerufen werden, welche die Referenzierung von Objekten aufzeigt. Die Anzeigeliste kann als

- Report oder als
- Objektliste

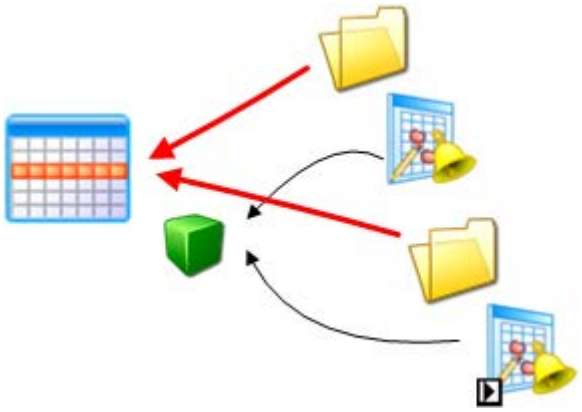
implementiert werden. Die Objektliste bietet den Vorteil, dass die aufgeführten Objekte vom Benutzer direkt angewählt werden können.

Vorgehen

1. Festlegen, welche Felder in der Verwendungsliste angezeigt werden sollen. In der Verwendungsliste werden die Parent-Objekte angezeigt, welche die Referenz enthalten. Hier ist zu beachten, dass sich die Objektklassentypen dieser Parten-Objekte stark unterscheiden können. Deshalb dürfen nur Attribute aus deren Schnittmenge als Listenfelder verwendet werden. Die Liste sollte sich auch wenige, generelle Felder beschränken.
2. Definieren, welche Referenzen in der Verwendungsliste aufgeführt werden sollen:
   Ein häufiger Fall, ist ein Nachweis über die Link- und Alias-Einträge. Eine andere Anwendung ist der Verwendungsnachweis von Objekten in den Attributen.
   Es muss auch festgelegt werden, wie die Darstellung von Referenzen ohne Berechtigung gelöst werden soll. Typischerweise werden diese nicht angezeigt.
3. Alle Prozessstatus definieren, in welchen die betreffende Aktivität zur Verfügung stehen soll. Oft möchte man die Funktion "überall" verwenden können. Dies ist nicht nur mit hohem Aufwand verbunden, sondern häufig auch nicht sinnvoll.
4. In diesem Zusammenhang muss auch festgelegt werden, wer in welchem Status die betreffenden Aktivität auslösen kann. Bewährt hat sich die Regel «nur wenn das Objekt sichtbar ist» oder «in Kombination mit der Methode *object.attributes.view()*» .
5. Definieren des Attributprofils für die Anzeige der Verwendungsliste.
6. Definieren des SQL-Statements für den Constraint/Filter, welches die gewünschten Resultate zurück liefert s. *Contraint - Filter (Seite 282).*
7. Aktivität mit Methode *Object.ConstraintObjList()* in einem Status erzeugen und austesten.
8. Entsprechender DB-Script erzeugen, um automatisiert die benötigen Aktivitäten in allen Prozessstatus einzubauen.

Beispiel: Referenz aller Links & Alias

Der Verwendungsnachweis wird als Objektliste implementiert und es werden alle Links und Alias auf das aktuelle Objekt angezeigt.



SQL-Statement

Das SQL-Statement besteht aus einem standardisierten Teil, welcher auch durch die Vorgaben für Constraint/Filter gegeben sind:

```
SELECT DISTINCT
decode(rownum, 990, 10000002, sys_object.obj_id),  /* nur
die ersten 990 Einträge anzeigen
decode(rownum, 990, ' weitere Resultate gefunden! ',

sys_objclass.super_img_colapsed),
sys_objclass.objclass_name, ss.type,
sys_object.classification
?attribList,      /* s. Hinweis
sys_objclass.sort_level

FROM
sys_object_tbl sys_object, sys_attribute, sys_objclass,
sys_structure ss

WHERE
sys_object.obj_id = ss.lower and sys_object.obj_id =
sys_attribute.obj_id and
sys_attribute.page_nr = 0 and    /* wichtig, damit immmer
korrekt
sys_object.objclass_id = sys_objclass.objclass_id and
ss.type = 'ORIGINAL'       /* nur das richtige Objekt
```

Das Suchresultat des Constraints muss auf die gewünschten Links und Alias Referenzen eingeschränkt werden

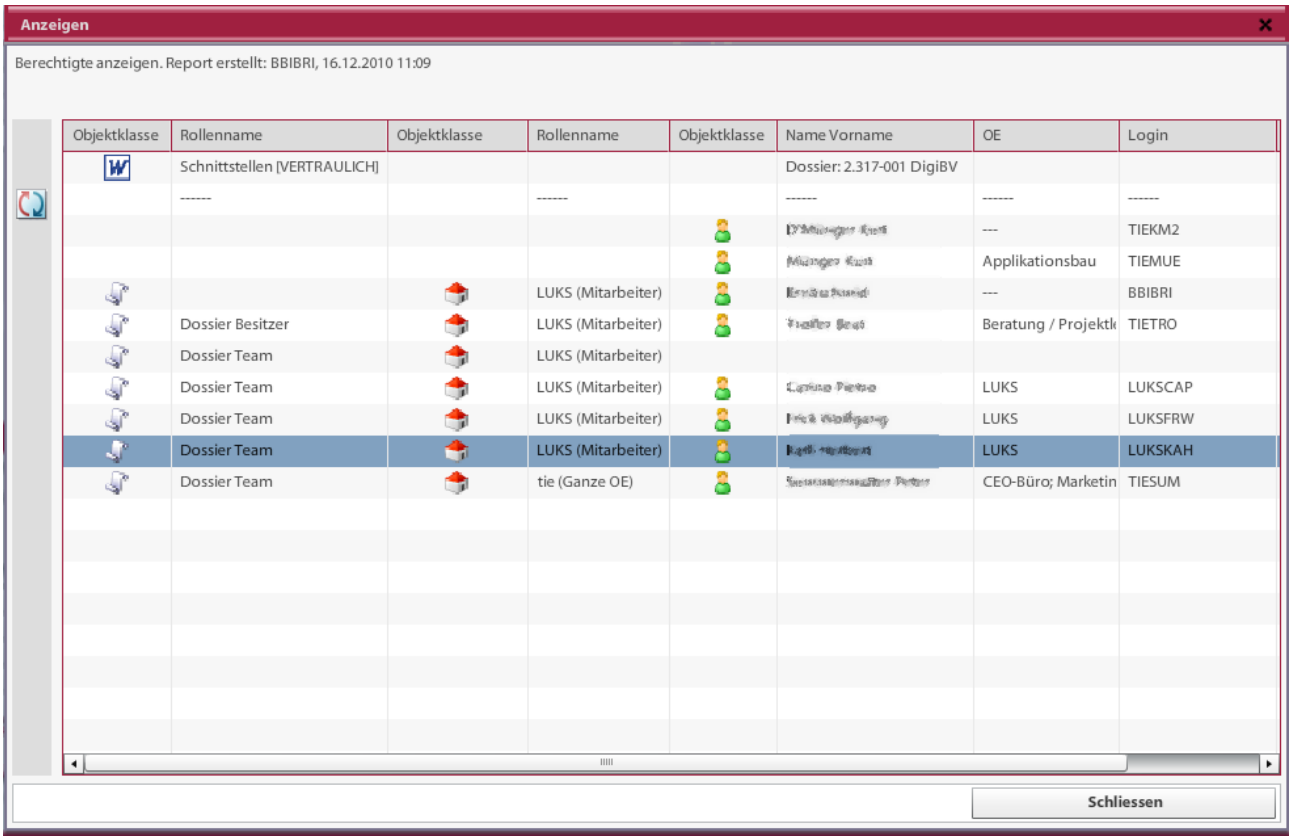
```
and sys_object.obj_id in
  (SELECT upper
   FROM sys_structure, sys_object_tbl
   WHERE
      type = 'LINK' and lower = ?obj_id  and
      upper = obj_id and
      objclass_id != 10080250
       /* bestimmte Objekte werden explizite ausgeschlossen
   UNION
      SELECT upper
      FROM sys_structure, sys_p_alias
      WHERE
        obj_id_org = ?obj_id and
        lower = obj_id and type = 'ALIAS'
  )
and rownum <= 990
```

Hinweise

- In der Liste werden nur die ersten 990 Einträge angezeigt. Gib es mehr Einträge, wird eine zusätzliche Zeile mit einem entsprechenden Hinweis angezeigt (Achtung: Sprachversionen)
- Das Objekte kann mehrere Attributseiten aufweisen, deshalb ist das Ergebnis auf die erste Seite einzuschränken. Dies empfiehlt sich auch dann, wenn im Moment nur 1 Seite verwendet wird.
- Bestimmte Objekte könne u.U. stören oder sollen nicht aufgeführt werden. Sie werden explizite ausgeschlossen.
- Der Parameter ?attribList wird vor Ausführung des SQL-Statements durch die Einträge im Attributprofil ersetzt. Dabei werden die Felder der BODY-Einträge in Reihenfolge der Sequenz in das SELECT Statement eingefügt.

Berechtigte anzeigen

Über die Anzeige der Berechtigen kann eine Liste abgerufen werden, die Zugriff auf ein klassifiziertes Objekt haben. Die Anzeigeliste wird als Report implementiert.



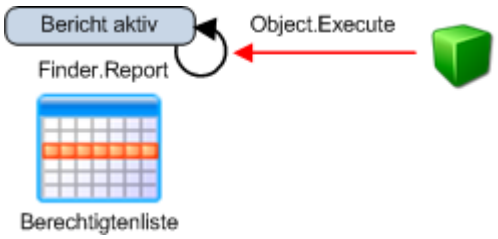
Vorgehen

1. Festlegen, welche Felder in der Berechtigtenliste angezeigt werden sollen. In der Berechtigtenliste werden die Benutzer-Objekte angezeigt, welche die Zugriff auf das klassifizierte Objekt haben.
2. Alle Prozessstatus definieren, in welchen die betreffende Aktivität zur Verfügung stehen soll.
3. Definieren des Attributprofils für den Report der Berechtigtenliste.
4. Report Ergebnismaske festlegen
5. Definieren des SQL-Statements für den Report, welches die gewünschten Resultate zurück liefert s. *Reports (Seite 284).*

   Beispiel SQL-Statement:

```
SELECT
   tbl.char4, tbl.char3, tbl.char6, tbl.char7,
tbl.char5,
   tbl.char2, tbl.char8, tbl.char1, tbl.num2
FROM
    table(C_RP_BERECHTIGTE(?caller_obj_id, ?obj_id, ''))
tbl
ORDER BY 9, 2, 3, 4, 6
```

6. Aktivität mit Methode *Object.Execute()* in einem Status erzeugen und austesten.



7. Entsprechender DB-Script erzeugen, um automatisiert die benötigen Aktivitäten in allen betroffenen Prozessstatus einzubauen.

Hinweis

- Die Parameter der Methode *Object.Execute()* sind wie folgt aufzusetzen:
  P1: Report ID
  P2: Aktivität-ID für *Finder.Report()*
  P3: Parent-ID von Report

Häufig gestellte Fragen

Was durchsucht die Volltextsuche?

Im Hintergrund werden neue Dokumente automatisch indexiert und werden beim Durchsuchen mit der Volltextsuche berücksichtigt. Was genau indexiert wird, wird an folgenden Stellen definiert:

- Objekt Metadaten / Attribute: wird in jeder Objektklasse festgelegt
- Dateien – wird in den Dateierweiterungen jeder Objektklasse festgelegt

Wie wird festgelegt, was beim Anklicken eines Objektes in der linken Baumanzeige geschehen soll?

Die Reaktion des Objektes wird durch eine Aktivität festgelegt, welche dem Objekt im aktuellen Prozessstatus zur Verfügung steht. Üblicherweise sind in einem Status mehrere Aktivitäten festgelegt, deshalb muss die gewünschte Klick-Aktivität in den Konfigurationsdaten speziell markiert werden. Weiter Erklärungen sind im Abschnitt «*Geschäftsprozess* (Seite 62) > Die Aktivität» zu finden.

Eine erwartete Funktion steht im Kontextmenü nicht zur Verfügung.

Die Auswahl der Kontextfunktionen wird durch die zugewiesenen Ausführungsrechte bestimmt. Als erstes ist zu prüfen, ob der gewünschten Aktivität für den betreffenden Benutzer die Ausführungsrechte erteilt worden sind – s. Abschnitt «*Aktivität: Zuweisung von Rollen* (Seite 96)».

Es kann auch sein, dass sich das Objekt nicht im vermuteten Prozessstatus befindet, und damit die erwartete Aktivität nicht definiert ist. Prüfen Sie die Tooltip-Anzeige des Objektes oder die Einträge in der Workflow-Liste, wo der effektive Status angezeigt wird. Eine direkte Anzeige des zugehörigen Prozesses ist nicht möglich.

Eine Methode kann in der Tabelle METHODE_RESTICTION eingetragen sein. In diesem Fall kann eine Aktivität mit dieser Methode nur verwendet werden, auch die aktuelle Objektklasse einen Eintrag in dieser Tabelle aufweist. Im Gegensatz zum Tabellenname müssen dort alle Objektklassen eingetragen werden, für die keine Restriktionen gelten.

Die definierten Aktivitäten eines Prozessstatus sind beim Parametrisieren plötzlich verschwunden.

Die Aktivitäten eines Status werden nur aufgelistet, wenn in der Objektklasse **ToDo** vorgängig das Feld **Typ** auf «ObjList» gestellt wurde.

Wie erreicht man höchst mögliche Sicherheit gegen unerlaubten Zugriff auf die Dokumente?

Voraussetzungen für einen sicheren Betrieb sind:

- eine richtig konfigurierte Firewall
- ein gutes Passwort-Management, Benutzer-Zertifikate
- Einsatz von verschlüsseltem Datenverkehr
- nur der Web-Server liegt in der DMZ, alle anderen System laufen auf Rechnern hinter einer zweiten Firewall
- sauber aufgesetzte Benutzerrollen
- restriktive Ausführungsberechtigungen für Aktivitäten
- Einsatz von eingeschränkten Rechten bei Methoden

Können die Daten auf dem Datenträger selbst auch verschlüsselt abgelegt werden?

Für höchste Sicherheitsanforderungen kann es notwendig werden, auch die Daten auf dem FTP-Server selbst zu verschlüsseln. Befindet sich der FTP-Server auf einem Windows 200x Rechner, so kann dazu das Microsoft Encrypting File System (EFS) eingesetzt werden. Es erlaubt eine gezielte Verschlüsselung pro Verzeichnis.

Wie wird die Dokument-Anonymisierung definiert?

Mit der Dokument-Anonymisierung können einzelne Bereiche von Office-Dokumenten gesperrt oder freigegeben werden. Die Festlegung erfolgt über zugeordnete Attributlisten. Dieses Feature ist in der aktuellen Software Version noch nicht verfügbar.

Ich habe ein neues Objekt erzeugt, aber es wird weder im Baum noch in der Liste angezeigt.

Ein Objekt wird dem Benutzer nur angezeigt, wenn er einer Rolle angehört, welcher das Sichtbarkeitsrecht erteilt wurde. Zudem muss sich das Objekt in einem Prozessstatus befinden, in welchem es mindestens eine Aktivität gibt, für welche der Benutzer das Ausführungsrecht hat.

Ich kann nicht alle Rollen Definitionen finden !

Wenn es sich um eine persönliche Rolle handelt, so wird diese unter *Hauptordner/_Mandant/Rolle* abgelegt. Systemrollen finden Sie unter *Hauptordner/_Paradef/Rollen*. Die Rollen für die Steuerung der Standardberechtigungen in OE-Ablagen, AGs oder Projektgruppen sind immer unter den entsprechenden

Ordnern im Verzeichnis Berechtigungen abgelegt. Die hierarchischen Rollen der Organisationsstruktur sind in der jeweiligen Hierarchiestufe gespeichert.

Warum sind Rollen und Prozesse für Projekte nicht auf den «normalen» Verzeichnissen abgelegt?

Bei Projekten (und auch bei AGs und OEs) sind die Rollen und Prozesse im Projekt angelegt. Dies dient nicht nur der besseren Übersicht, sondern ermöglicht eine einfache Individualisierung des Projektes. In solchen Anwendungsfällen wird man immer eine/mehrere Benutzerrollen benötigen, um die Teilnehmerrechte verwalten zu können. Die direkte Anordnung der Rollen beim Projekt macht deshalb Sinn.

Prozess und Benutzerrollen sind direkt miteinander verknüpft. Wenn individuelle Rollen verwendet werden, braucht das Projekt deshalb auch einen individuellen Satz von Prozessen – auch wenn diese nicht abgeändert werden müssen. Mit der Speicherung der Prozesse unter dem Projekt bleibt die notwendige Übersichtlichkeit erhalten.

Welche Arbeiten muss der Systemverwalter beim Eintritt eines neuen Benutzers in das Unternehmen durchführen?

Der Mitarbeiter wird als neuer *i-engine* Benutzer erfasst. (s. Benutzerhandbuch > Benutzer Verwalten).

In der Organisationsstruktur wird der Benutzer der hierarchischen Rolle zugeordnet. Damit erhält er automatisch das Leserecht auf alle OE-Ablagen, Projekte und AGs, denen diese Rolle zugewiesen wurde.

Für alle weitergehenden Zugriffsrechte auf Ordner muss der Benutzer jeweils den zuständigen Benutzerrollen zugewiesen werden. z.B. Mitgliedschaften bei Arbeitsgruppen, Schreibrechte auf der Abteilungsablage etc.

Der volle Zugriff auf die persönlichen Ordner ist automatisch erteilt.

Ich möchte für bestimmte Benutzergruppen ein Lese- und Schreibrecht auf Dokumente erteilen?

Eine solche Zugriffssteuerung muss in der Anwendung aufgebaut werden. *i-engine* bietet 3 Grundrechte:

Zugriffsrecht: Ein Objekt kann verwendet werden, z.B. als Vorlage.

Sichtbarkeit: Ein Objekt wird in einer Auflistung angezeigt.

Ausführbarkeit: Eine Aktivität kann ausgeführt werden.

Um ein Lese-/Schreibrecht zu implementieren, wird man wie folgt vorgehen:

In einem solchen Fall wird für die betreffenden Objekte (Ordner/Dokumente) das Zugriffs- und Sichtbarkeitsrecht für alle Benutzer erteilt. Die Aktivitäten werden in

«Lesefunktionen» und «Schreibfunktionen» eingeteilt. Es wird eine Leseberechtigungs-Rolle und eine Schreibberechtigungs-Rolle erstellt, und den beiden Funktionsgruppen zugeteilt. Die Benutzer werden nun entsprechend ihrem Lese- resp. Schreibrecht als Mitglieder in den beiden Rollen eingetragen.

Wie lässt sich festlegen, welche Vorgänge in der Workflow-Anzeige protokolliert werden?

Prinzipiell kann die Ausführung jeder Methode als Vorgang in der Workflow-Anzeige eingetragen werden. Wenn eine Methode zu einem neuen Prozessstatus führt, erfolgt immer ein Eintrag. Findet keine Statusänderung statt (z.B. Abfrage von Eigenschaften), so steht es dem Parametriker frei, auf eine Eintragung zu verzichten.

Um die Protokollierung in diesem Fall auszuschalten, muss als nächster Prozessstatus bei der betreffenden Aktivität **Keine Statusänderung** gewählt werden.

Welche Möglichkeiten gibt es, die Anzeige im Strukturfenster festzulegen?

Prinzipiell wird die Anzeige im Strukturfenster durch das Anklicken einer Direktverknüpfung ausgelöst. D.h. weder über Klicks im Listenfeld noch eines Kontextmenüs kann die Anzeige im Strukturfenster beeinflusst werden.

Über die Direktverknüpfungen wird die Anzeige des Inhaltes des zugeordneten Hauptordners ausgelöst. Sämtliche Objekte in diesem Ordner werden als hierarchische Struktur dargestellt. Allerdings erscheinen nur Objekte, deren Eigenschaft **Eintrag im Baum** angewählt wurde – und für die der Benutzer natürlich auch die notwendigen Berechtigungen besitzt.

Weitgehende Möglichkeiten zum Realisieren von Menüstrukturen bietet die Verwendung von **Executor** Objekten. Damit lassen sich beliebige interaktive Menüs im Strukturfenster ausbauen. D.h. beim Anklicken kann eine beliebige Web-Aktivität ausgelöst werden (nicht nur der Listenaufbau im Listenfeld).

Die i-engine Funktionen stehen mir in der Office-Umgebung nicht zur Verfügung.

Bei jedem Aufstarten von *i-engine* wird geprüft, ob sich die notwendigen Erweiterungen bereits auf dem Arbeitsplatzrechner befinden. Falls sie noch nicht vorhanden sind, werden sie automatisch auf den Rechner kopiert und die Funktionen sollten Ihnen zur Verfügung stehen. Wenn Sie trotzdem die Funktionen in Menü nicht zur Verfügung haben, so kann das an den Einstellungen der Office Software liegen (Macro Sicherheitseinstellung) oder weil anderweitige, nicht-kompatible Software auf Ihrem System installiert ist. Wenden Sie sich an Ihren Systemverantwortlichen zur Abklärung.

Hinweis: Die SAP-Client Software und *i-engine* sind nicht kompatibel.

Bevor Sie die *i-engine* Funktionen in MS Office verwenden können, müssen Sie *i-engine* mindestens ein Mal gestartet haben. Das *i-engine* Add-in für Excel muss manuell eingeschaltet werden: Dazu im Menü **Tools | Add-In** das *i-engine* Add-in anwählen.

Anhang

Merkblatt Nomenklatur

Oracle Bezeichner	AF, BC, DY, NX, PA, RO <Custom>_<SQL Typ>_<Modul>_<Prozess>_<Status>_<Aktivität> CB, DE,PO, VA <Custom>_<SQL Typ>_<Modul>_<Prozess>_<ProfilRef> CS, TA <Custom>_<SQL Typ>_<Modul>_<Prozess>_<Status>_<IPC> CN, FN <Custom>_<SQL Typ>_<Modul>_<Prozess>_<Finder> EX, FK, RE, RT <Custom>_<SQL Typ>_<Modul>_<Prozess>_<Status>
SQL-Statements	AF, BC, DY, NX, PA, RO <Prozess>_<Status>_<Aktivität> CB, DE,PO, VA <Prozess>_<ProfilRef> CS,TA <Prozess>_<Status>_<IPC> CN, FN <Prozess>_<Finder> EX, FK, RE, RT <Prozess>_<Status>
PL/SQL Routinen	identisch wie zugehörige Oracle Routine
Attribuprofil	<Modul>_<Objektklasse>_<Methode> <Modul>_<Objektklasse>_<Methode>_<Status>

Feld	Max	Werte resp. Dokumentiert in
<Custom>	2	«CF» Function / «CP» Procedure «TF» Template Function / «TP» Template Procedure
<SQL Typ>	2	Entsprechend dem SQL-Statement Typ aufgesetzt
<Modul>	3	Dem *Verzeichnisname* des Moduls wird das Modul Mnemonic vorangestellt. Standard Module: «HE» health / «BE» business / «PE» public / «IE» generell
<Prozess>	3	Im Feld *Beschreibung* des Prozesses wird das Prozess Mnemonic vorangestellt.
<Status>	3	Im Feld *Beschreibung* des Prozessstatus wird das Status Mnemonic vorangestellt. Einfachheitshalber eine Nummer.
<Aktivität>	9	Im Feld *Beschreibung* der Aktivität wird das Aktivitäts Mnemonic vorangestellt.
<ProfilRef>	12	Der *Name* des Attributprofils ist strukturiert aufgebaut, und ist zu lang für die Verwendung in anderen Bezeichnern. Deshalb muss im Feld *Beschreibung* des Attributprofils ein zusätzlicher Mnemonic festgehalten werden.
<IPC>	9	Falls der IPC Namen nicht eindeutig ist oder die maximale Länge überschritten wird, wird das IPC Mnemonic dem effektiven Namen vorangestellt.
<Finder>	12	Im Feld *Beschreibung* des Finders/Constraints/Reports wird das Mnemonic vorangestellt. Falls der *Objektname* eindeutig ist und die maximale Länge nicht überschreitet, ist keine weitere Dokumentation erforderlich.
<Methode>		Name der Methode, wobei der Namensteil «Object» weggelassen wird.
<Objektklasse>		Name der Objektklasse

Hinweis

- Bei Bezeichnern für Attributprofile wird für <Methode> und <Objektklasse> die Grosskleinschreibung verwendet.
- Bezeichner enthalten keine Leerzeichen. Diese werden einfach weggelassen.
- PL/SQL Templates: <TF>_<SQL Type>_<Methode>
- Standard Attributprofile: IE_Default_<Methode>_<Anwendung>

Applikations- und wichtige System-Objektklassen

Die Felder der Objektklasse DOCUMENT sind die Basis für alle Objektklassen.

DOCUMENT	
OBJECT_TBL	OBJ_ID
	OBJ_NAME - sowie OBJ_NAME_Lx
	DESCR - sowie DESCR_Lx
	LAST_UPDATE - Status protokolliert
	CREATION_DATE
	LOCATION_ID - physische Ablage
	CLASSIFICATION
	ENCRYPTION
	OBJ_INDEX - bei Versionierung
	ROLE_ID - Besitzer des Objektes
	USER-NAME - ausgecheckt Bearbeiter
	SIZE_xxx_FILE - Dateigrösse
	FOLDER_FILE_NAME
	LOCAL_FILE_MA;E
	CLIENT_FILE_NAME
ATTRIBUTE	PAGE_NR
	T1..T20; D1..D10; N1..N20
	Un_HREF, Un_DISP_Lx - Web-Link
ATTRIBUTE_LONG	LONG1..10
OBJ_STATUS	EXPIRATION_DATE - aus WF_STATUS
	NOTIFICATION_DATE - aus WF_STATUS
	WF_NOTE - Tooltip Anzeige
	NEXT_EXPIRATION_DATE
	NEXT_NOTIFICATION_DATE
WF_STATUS	STATUS_NAME - sowie STATUS_NAME_Lx
	DESCR
NOTES	TITLE
(mehrere Records)	CONTENT
	USER_NAME
	LAST_UPDATE
OBJCLASS_TBL	OBJCLASS_NAME - sowie .._Lx

	OBJCLASS_DESCR
EXTENSION	EXTENSION_TYP
	DEFAULTOPENER «YES» Extension anzeigen «NO»

Bemerkungen

- Die Dateigrösse wird separat für Application, View und Print Datei abgelegt. Allerdings wird die Grösse nur beim Einchecken bestimmt. Bei automatisierten Prozessen ist der Wert nicht aufgesetzt.
- Der Inhalt der NOTES Records wird auf dem Notizregister angezeigt. Dazu wird im Attributprofil die Funktion */method/jsplib72/notes_edit.jsp?txnid* eingesetzt.
- WF_NOTE wird in der Listenanzeige auf dem Prozessstatus-Symbol als Tooltip angezeigt. s. auch Ausführungen zum Feld *Statusnotiz* in der Beschreibung zu «Aktivität».
- Während «EXPIRATION_DATE» ggf. das Datum aus der letzten Aktivität enthält, entspricht «NEXT_EXPIRATION_DATE» dem Datum der aktuellen *Distribute()* Aktivität.
- In WF_STATUS ist die Zeitdauer für Expirtaion/Notification festgehalten. In OBJ_STATUS sind die resultierenden absoluten Endzeitpunkte definiert.

USER	
P_USER	USER_NAME
	FULLNAME
	ROLE_ID
	EMAIL1..3
	MOBILE - SMS
	USER TYPE «USER» «SERVICE MANAGER» «IMPORTER»
Patientenprozess	HL7_DOCTOR_ID - Geschäftspartner-ID
	ASAS_ID - HIN Anmeldung
	EAN
	DOCBOX_ID

Bemerkungen

- Für den Patientenprozess sind weitere Daten definiert:
  HL7 User Idenifikation
  Anmeldeinformation für das Health Info Network (HIN)
  Zugeordneter EAN-Code
  Anmeldeinformation für Docbox Datenaustausch

ROLLE	
ROLLE	ROLE_NAME - sowie ROLE_NAME_Lx
	ROLE_TYPE: «PRIVATE» «PERSONAL» «SYSTEM» «OU_EMP» «OU_DOWN» «OU_PROXY» «OU_UP» «OU_BOSS» «USER_DEFINED» «PR_USER_DEFINED»
	OU_ID
	RANKING - Sortierung des Typs «PR_USER_DEFINED»

PATIENT	
PATIENT	PATIENT_ID
	PID
	PATIENTS_FAMILIY_NAME
	PATIENTS_GIVEN_NAME
	PATIENTS_ADDRESS_STREET
	PATIENTS_ADDRESS_ZIP_CODE
	PATIENTS_ADDRESS_DOMICILE
	PATIENTS_SEX
	PATIENTS_BIRTH_DATE
	MAIDEN_NAME
	COUNTRY
	CANCELATION_DATE - Storno
	DEATH_DATE

VISIT	
VISIT	VISIT_ID
	FID
	PID
	VISIT_TYPE
	ADMISSION_DATE - Eintritt
	DISCHARGE_DATE - Austritt
	FINANCIAL_CLASS - Kostenstelle
	POINT_OF _CARE - Pflege OE
	FACILITY - Fach-OE

STUDY	
STUDY	PID
	FID
	ACCESSION_NUMBER - Geräte-FID
	STUDY_INSTANCE_UID - Gerätenummer
	STUDY_DESCRIPTION
	STUDY_DATE_TIME
	STUDY_IDENTIFICATION
	REFERING_FAMILIY_NAME
	REFERING_GIVEN_NAME

Bemerkung

- Die Fall-ID wird u.U. erst mit den HL7 Daten angeliefert. In diesem Fall wird temporär die vom Gerät vergebene ACCESSION_NUMBER als FID verwendet.

OBJECT_REF	
OBJECT_REF	OBJ_ID
	TYPE «CONSULTING DOCTOR» «ATTENDING DOCTOR» «REFERING DOCTOR» «ADMITTING DOCTOR»
	SEQ - Reihenfolge wenn mehrere
	REF_OBJ_ID - Referenziertes Objekt

Bemerkung

- Hilfstabelle, mit welcher sich 1:n Relation abbilden lassen.
- Im Patientendossier wird damit die Zuordnung Patient zu Arzt festgehalten. Die Tabelle wird durch den HL7-Importer nachgeführt.

OBJECT_SHOW	
OBJECT_SHOW	OBJ_ID
	ROLE_ID
	ACCESS_LEVEL - 0..999
	SHOW_DATE
OBJECT_ACCESS	
OBJECT_ACCESS	OBJ_ID
	ROLE_ID
	ACCESS_LEVEL - 0..999
	ACCESS_DATE

Bemerkung

- Über einen Eintrag wird das Startobjekt definiert, ab welchem die Rolle Lese resp. Schreib-Zugriff hat. Das Zugriffsrecht wird im Objektbaum nach unten vererbt.
- Die Vererbung nach unten kann durch einen RESTRICT Eintrag - s. nächste Tabelle - gestoppt werden.
- Damit der Zugriff auf den aktuellen Ordner möglich ist, wird bei der Erteilung von Schreibrechten in allen übergeordneten Objekten – die sich im Pfad zum Wurzelobjekt befinden - die Sichtbarkeit für die betreffende Rolle automatisch ebenfalls erteilt. Andere Objekte, die sich in solchen automatisch freigegeben Strukturelemente befinden, werden dadurch nicht ebenfalls sichtbar. Der Zugriffspfad kann deshalb für den Benutzer durch Ordnerstufen ohne Inhalt führen.

RESTRICTION_SHOW	
RESTRICTION_SHOW	OBJ_ID
RESTRICTI-ON_ACCESS	
RESTRICTION_ACCESS	OBJ_ID

Bemerkung

- Die automatische Vererbung der Rechte auf untergeordnete Ordnerstufen kann jederzeit unterbrochen werden. Dazu wird in der Berechtigungsstruktur eine «Restriktion» definiert, die die Vererbung der entsprechenden Rechte nach unten und nach oben unterbindet.

Tabellen in i-engine

Tabellen für Basisobjekte

Hier die wichtigsten Tabellen für Basisobjekte

ATTRIBUTE_PROFILE	Definition der Maskenfelder
ATTRIBUTE	Werte der Maskenfelder einer Objekt-Instanz
ATTRIBUTE_LONG	Lange Textstring der Maskenfelder einer Objekt-Instanz
EXTENSION	Dateierweiterungen einer Objektklasse
IPC	Festlegungen für die Interprozess Kommunikation
OBJCLASS	Definition der Objektklassen
OBJECT	Instanz eines Objektes
OBJ_STATUS	Aktueller Prozessstatus eines Objektes
OBJECT_ACCESS	Zugeordnete Rollen für Zugriffsrecht
OBJECT_SHOW	Zugeordnete Rollen für Sichtbarkeit
RESTRICTION_ACCESS	Unterbruch Vererbung Zugriffsrechte
RESTRICTION_SHOW	Unterbruch Vererbung Sichtbarkeit
OBJECT_PROPERTY	Zuordnung zwischen Attributprofil und Objektklasse
WF_PROCESS	Definition Workflow - Prozesse
WF_STATUS	Definition Workflow - Prozessstatus
WF_ACTIVITY	Definition Workflow - Prozessaktivität
WF_ACTIVITY_PROFILE	Abweichungen der Zuordnung von Attributprofil/Objektklasse
WF_DECISION	Dynamische Prozessablaufsteuerung, bedingte Prozessstatus-Übergänge

Zusatztabellen für Objektklassentypen

APPOINTMENT	Exchange Termin-Objekte
CLIENTFILE	Zusatzdateien für AutoUpdate
FINDER	Constraint und Finder Objekte
IMAP	IMAP Protokoll Definitionen
MAIL	Mail Objekte

MANDANT	Mandanten Objekte
ORGANIZATIO- NAL_UNIT	OU Objekte
P_ALIAS	Alias Objekte
P_RECYCLEBIN	Papierkorb Objekt
P_USER	Systembenutzer Objekte
ROLE	Rolle Objekte
ROLE_CONTENT	Rolle Objekte
STYLESHEETS	CSS Dokument-Objekte
TASK	Exchange Aufgaben-Objekte
WEBDAV	WebDAV Protokoll Definitionen

Hilfstabellen

BOILERPLATE	Meldungstexte
LOG	Fehlerprotokoll
P_CODE	Texte, Feldbeschreibungen
PROPERTIES	Diverse ergänzende Informationen. z.B. Zuordnung Menüpunkt und Root-Objekt Definition WebDAV Server
STRUCTURE	Definition der Baumanzeige für Root-Objekte
SQL_STMT	SQL-Statements
WATCH_LIST	E-Mail Abonnemente
WATCH_PROCESS	Zuordnung zwischen Aktivität und EMAIL

Applikation Symbole für Kundenprojekte

Für die Markierung von Objektklassen steht eine umfangreiche Sammlung zur Verfügung. Dabei ist jedes Symbol ausser in der Grundform auch in einer Vielzahl von erweiterten Darstellungen vorhanden. Die Symbole werden durch die Referenz-ID definiert;
bspw. «fB2.01» -> Einkaufskorb mit Pluszeichen in der "Finanz" Sammlung.

Symbol Overlays

x.00	x.01	x.02	x.03	x.04	x.05
x.06	**x.07**	**x.08**	**x.09**	**x.10**	**x.11**
x.12	**x.13**	**x.14**	**x.15**	**x.16**	**x.17**
x.18	**x.19**	**x.20**	**x.21**		

Diese Overlays können auf alle nachfolgenden Symbole angewandt werden.

Thema Finanzen

f	1	2	3	4	5	6
A						
B						
C						
D						
E						
F						
G						
H						
J						
K						
L						
M						

Thema Business

b	1	2	3	4	5	6
A						
B						
C						
D						
E						
F						
G						
H						
J						

Thema Generell

g	1	2	3	4	5	6
A						
B						
C						
D						
E						
F						
G						
H						
J						
K						
L						
M						
N						
O						

Thema Networking

n	1	2	3	4	5	6
A						
B						
C						
D						
E						
F						
G						
H						
J						
K						

Thema Healthcare

h	1	2	3	4	5	6
A						
B						
C						
D						
E						
F						
G						
H						
J						
K						
L						
M						

Glossar

A

Ablage

Die Dokument-Dateien werden üblicherweise auch in der Datenbank abgelegt. i-*engine* unterstützt für die Ablage der Dokumentobjekte neben der normalen Datenbank-Speicherung weitere Speichertechniken wie FTP-Server (externe Dateiablage), XDB via FTP (Datenbank), ORDDICOM Dicom-Bilder (Datenbank), BLOB (Datenbank), URI (lokales Filesystem oder NFS), SMB (Samba Fileshare) oder IBM DR550 (Archivmedia).

Aktivität

Abhängig vom Zustand eines Dokumentes sind weitere Arbeitsschritte notwendig resp. möglich. Diese Aktivitäten (Methoden, Arbeitsprozesse oder Bearbeitungsfunktionen) sind Einzelschritte des gesamten Geschäftsprozesses z.B. "Modifizieren", "Visum erteilen", "Verteilen". Wenn die Aktivität beendet ist, hat das Dokument wieder einen definierten Zustand erreicht.

Die möglichen Aktivitäten aus jedem Zustand heraus werden vom Prozessverantwortlichen definiert. Dazu steht ihm ein umfassendes Sortiment von Grundoperationen zur Verfügung.

Attribut

Die Attribute resp. Metadaten sind ein Teil der Eigenschaften eines Dokumentes (Objektes). Jedem Objekt können beliebige Attributfelder zugeordnet werden. Dies kann z.B. der Name des Erstellers, die Anzahl Rechnungspositionen oder der Budgetbetrag eines Projektes sein. Die Anzahl und Art der Attribute wird für die jeweilige Anwendung und Dokumentenklasse festgelegt – kann aber jederzeit angepasst werden. Die Attributfelder und -werte können zur Klassierung und Suche eingesetzt werden.

Attributprofil

Der Aufbau von Dialogmasken wird über das Attributprofil festgelegt.

B

Basis-Objektklassen

In der Grundausstattung enthält *i-engine* für jeden Objektklassentyp eine entsprechende Objektklasse. Nur in Ausnahmefällen müssen die Basis-Objektklassen ergänzt werden.

Berechtigungsstufe

In der Berechtigungsstruktur wird eine Zuordnung zwischen Rollen einerseits und Objekt- und Ausführungsrechten andererseits festgelegt. Alle Objekte sind in die Berechtigungsstruktur eingeordnet und erhalten dadurch je einen Stufenwert für das Zugriffs- und das Sichtbarkeitsrecht für jede Rolle.

Das jeweilige Objektrechte ist erteilt falls der Stufenwert grösser Null ist, wobei die Werte vererbt werden.

Für jede Aktivität kann die minimal erforderliche Berechtigungsstufe für die Ausführung zugeordnet werden. Der Benutzer erhält entweder über die Position des Objektes in der Berechtigungsstruktur oder aber dynamisch durch den Workflowprozess seinen aktuellen Zugriffsrechtewert. Nur wenn der minimale Wert aus der Aktivität erreicht wird, kann die Aktivität ausgelöst werden.

D

Datenwächter

Über den Wächter kann dem Benutzer direkt ein Email zur Orientierung übermittelt werden, wenn sich etwas am Inhalt eines Ordners ändert. Welche Ablage-, Eingangs- oder Arbeitsordner durch den Benachrichtigungs-Assistenten beobachtet werden soll und welche Veränderungen - z.B. neues Protokoll abgelegt - eine Meldung auslösen sollen, kann frei definiert werden. So könnte z.B. der Betreffende jeweils ein Email erhalten, wenn ein neues Dokument visiert werden muss.

Dokument

Die Grundlage der enormen Flexibilität von *i-engine* ist die Objekttechnologie. Damit kann das System mit beliebigen Typen von Geschäftsobjekten umgehen: Rechnungen, Briefe, Projektordner, Personendaten, Konstruktionszeichnungen oder Akten. Alle verwalteten Objekte werden generell als Dokumente bezeichnet.

Dokumentarten

Innerhalb einer Objektklasse können ggf. unterschiedliche Ausprägungen eines Objektes verwendet werden. So kann z.B. in der WinWord Objektklasse unterschiedliche Arten von Dokumenten geben, wie Brief, Protokoll, Materialbestellung, ... Den unterschiedlichen Dokumentarten entsprechen unterschiedliche Dokument-Vorlagedateien bei der Erzeugung eines Objektes.

Dokumenteigenschaften

Zu jedem Dokument (Objekt) lassen sich die Eigenschaften aufrufen. Die Eigenschaften umfassen folgende Teile:

- Attributfelder: Metadaten
- Workflow: Historie
- Notizen: Bemerkungen
- Wächter: Überwachungsfunktionen

E

Extension (Dokumenttyp)

Jeder Objektklasse können ein oder mehrere Dokumenttypen zugeordnet werden. Falls es die Anwendung erfordert (z.B. bei CAD-Zeichnungen), können auch mehrere unterschiedliche Dokumenttypen einer Klasse zugewiesen werden. Der Typ wird über die Extension charakterisiert. Die Information aus dem Dokumenttyp wird u.a. für die Bestimmung der Drittanwendung für die Bearbeitung benötigt.

K

Klassifikation

Über die Klassifikation werden die Dokumente und Dateien in einer – oder mehreren – hierarchischen Struktur(en) geordnet. Die Klassifikation kann über spezielle Strukturobjekte oder über Metadaten festgelegt werden.

Die Klassifikation des Objektes ist unabhängig von der physischen Ablage oder den Zugriffsberechtigungen. Allerdings wird in der Praxis die Zugriffsberechtigung und Klassifikationsstruktur bei einfacheren Lösungen oft zusammengelegt.

L

Leitdokument

i-engine unterstützt die hierarchische Strukturierung der Dokumente. Sie können einem Hauptdokument weitere Dokumente zuordnen. Damit können z.B. Anhänge einem Vertrag, Funktionsbibliotheken einem Projekt oder E-Mails einem Vorgang zugewiesen werden. Welche Unterdokumente in wie vielen Hierarchiestufen einem Hauptdokument zugeordnet werden können, ist allerdings von der jeweiligen Installation abhängig.

O

Objektklasse

Alle Objektklassen weisen einen bestimmten Objektklassetyp auf und erben von diesem ihre Eigenschaften. In der Grundausstattung enthält *i-engine* eine Sammlung von Systemobjektklassen für jeden Objektklassentyp.

Applikations-Objektklassen werden üblicherweise aus den beiden Objektklassetypen «Document» oder «ObjList» abgeleitet.

Zur einfacheren Verwaltung der unterschiedlichsten Dokumente werden diese nach Objektklassen zusammengefasst. Die Festlegung von Klassen erfolgt vor allem in Abhängigkeit von den notwendigen Geschäftsprozessen und weniger nach der Dokumentart. Die Objektklassen werden im Konfigurationsmodus von *i-engine* definiert. Durch eine eigenen Objektklasse lassen sich auch unterschiedliche Symbole und Bezeichnungen zuordnen.

Objektklassentyp

Die Grundelemente der Objekthierarchie in *i-engine* sind die Objektklassentypen. Der Objektklassentyp bestimmt die Attribute und Methoden. Via Objektklasse werden die Eigenschaften des Objektklassentyps dem Objekt vererbt. Bei den Objektklassentypen handelt es sich um «abstrakte Klassen» (OO-Nomenklatur), d.h. es können keine weiteren Objekte daraus hervorgehen.

Es können keine neuen Objektklassetypen erzeugt werden.

Ordner

Für die Gliederung der Dokumentenablage werden wie gewohnt Ordner verwendet. Allerdings kann der Zusammenhang zwischen den *i-engine* Gliederungsordnern und dem eigentlichen Speicherort beliebig festgelegt werden. Damit ist die Datenspeicherung vom logischen Zugriff und der Strukturierung/Klassierung losgelöst – und kann entsprechend einfach optimiert werden.

P

Protokoll

Über die Eigenschaften jedes Objektes kann eine zeitliche Auflistung aller ausgeführten Aktivitäten abgefragt werden. Einträge in die Protokollliste sind über den «Nächsten Status» Eintrag in der Aktivität gesteuert. Als Empfehlung gilt, dass Aktivitäten mit Datenveränderung protokolliert werden. Bei reiner Anzeige ist eine Protokollierung oft unnötig, da dem Benutzer ja die entsprechende Berechtigung ja erteilt wurde.

Prozess

Der Prozess legt den Arbeitsfluss für eine bestimmte Objektklasse fest. Ein Prozess besteht aus einer Abfolge von seriellen und parallelen Teilaktivitäten. Zwischen den Teilaktivitäten liegt jeweils ein definierter Zustand, den das Dokument annimmt. Durch die Kombination von Prozesszuständen und Aktivitäten wird der Arbeitsfluss für eine Objektklasse festgelegt. Der gesamte Ablauf wird durch den Prozessverantwortlichen festgelegt.

Prozessstatus

Während eines Geschäftsprozesses durchläuft das Dokument verschiedene Zustände z.B. "Erstellt", "Warten auf Visum", "Freigegeben". Jeder vorgesehene Zustand wird vorgängig beim Definieren des Prozesses festgelegt.

R

Rolle

Die Verwaltung von Benutzerrechten kann in grösseren Organisationen recht aufwändig sein. Unter Umständen müssen ganzen Gruppen von Benutzern identische Privilegien und Rechte für eine Vielzahl von Ordnern und Prozessen zugeteilt werden. Um dies zu erleichtern, kennt *i-engine* das Rollenkonzept.

Privilegien werden nicht einzelnen Benutzern sondern an Rollen bzw. Gruppen vergeben. Die Möglichkeiten, die sich durch die Benutzung von Rollen oder Gruppen ergeben, sind gleich. Allerdings ist die Philosophie, die hinter diesen beiden Begriffen steckt, unterschiedlich: Während *Rollen* Zusammenfassungen von Privilegien darstellen, sind *Gruppen* Zusammenfassungen von Benutzern. Die Rolle *Rechnungswesen* könnte alle Privilegien enthalten, die für die Bearbeitung von Rechnungen notwendig sind. Diese Rolle kann dann einzelnen Benutzern gewährt werden. Die Gruppe *Rechnungswesen* enthält dagegen die Benutzer, welche die Rechte für das Rechnungswesen benötigen. Die Effekte bei beiden Konzepten sind identisch.

S

Sicherheitsklassierung

Objekte und Dokumente können mit einer Sicherheitsklassierung versehen werden. Damit kann die Verwendungsmöglichkeit gesteuert und ggf. auch eine krypotgraphische Verschlüsselung der Informationen ausgelöst werden.

Die Sicherheitsklassierung wird vor allem bei Rollen-Objekten zur Einschränkung von Stellvertreterrollen und bei Dokumenten zur Verschlüsselung eingesetzt.

Standard Objektklassen

Alle im Customizing erstellten Objektklassen sind vom Objektklassentyp «Document» resp. «ObjList». Im Auslieferumfang von *i-engine* sind eine Reihe üblicher Objektklassen als Standard bereits erhalten

Strukturelemente

Strukturelemente werden eingesetzt, um Klassifikationshierarchien, Organisationsstrukturen oder andere hierarchische Informationen festzulegen und festzuhalten. Strukturelemente enthalten nur Metadaten und keine hinterlegten Dateien. In vielen Fällen dienen die Strukturelemente als Container (Ordner) für andere Objekte.

Systemklassen

Systemklassen sind Klassen, die aus den Objektklassetypen von *i-engine* abgeleitet werden. Systemklassen wird für jene Objekte verwendet, welche die Funktionen von *i-engine* implementieren. Im Gegensatz dazu sind die Applikationsklassen, welche kundenspezifische Objektklassen implementieren.

U

Unterdokument – gebunden/ungebunden

Es werden zwei Arten von Unterdokumenten unterschieden. Die verbundenen Unterdokumente werden zusammen mit dem Hauptdokument als eine Einheit verwaltet. Beide Dokumente befinden sich immer im gleichen Status und werden vom gleichen Ablaufprozess gesteuert. Es wird nur eine zweistufige Hierarchie unterstützt.

Die ungebundenen Unterdokumente sind vollständig unabhängige Dokumente, welche auch durch eigene Prozessabläufe gesteuert werden. Die Synchronisierung zwischen den Haupt- und Unterdokumenten muss durch entsprechende Prozesse erzeugt werden. Es wird eine beliebig tiefe Hierarchie unterstützt.

V

Verlauf

Im Verlauf werden die letzten 300 Objekte aufgeführt, für welche eine Aktivitäten ausgeführt wurde. Damit hat der Benutzer direkten Zugriff auf die zuletzt bearbeiten Objekte. Bei Mehrfacheinträgen erscheint nur der letzte. Im Gegensatz zum Protokoll wird im Verlauf nur festgehalten, dass etwas stattgefunden hat. In den Grundeinstellungen wird für jede Methode definiert, ob bei deren Ausführung ein Eintrag im Verlauf vorgenommen werden soll. Der Verlauf wird über die persönliche Baumstruktur angezeigt. Die Sortierung erfolgt nach Objektklasse.

VIP Durchbruch

Im Spitalbereich sind sog. VIP-Daten vor dem allgemeinen Zugriff geschützt. Trotzdem muss in Notfallsituationen die Möglichkeit bestehen, auf die erforderlichen Daten zugreifen zu können. Mit dem VIP-Durchbruch wird die Einschränkungen der Berechtigung für eine bestimmte Zeit durchbrochen. Allerdings werden die Vorgänge detailliert protokolliert und einem Verantwortlichen zur Kenntnis gebracht.

Vorlagen

Der Begriff Vorlage wird in unterschiedlichen Zusammenhängen verwendet.

Die **Systemvorlage** enthält die Einstellungen der Objektklasse. Sie kann zusätzlich auch eine Dokumentenvorlage enthalten. Beim Erstellen eines neuen Objektes dieser Objektklasse wird eine Kopie des Systemvorlagen Objektes – und falls definiert – auch der hinterlegten Dokumentvorlage gemacht.

Ein **Vorlagenobjekt** ist eine Instanz einer bestimmten Objektklasse. Das betreffende Objekt kann als Container beliebige andere Objekte (z.B. Dokumentobjekte) enthalten. Wird dieses Objekte als Vorlage für ein neues Objekt verwendet, werden zusätzlich auch Kopien aller enthaltenen Objekte angelegt.

Für das Erzeugen eines neuen Dokumentes wird eine Vorlage verwendet. In einer Objektklasse wird die Dokumentvorlage als integrierter Bestandteil der Systemvorlage für die betreffende Dokumentklasse festgelegt. Bei Vorlagenob-

jekten ist die **Dokumentvorlage** ein eigenständiges Objekt innerhalb des Vorlagenobjektes – der natürlich eine eigene Objektklasse zu Grunde liegt.

Index

A

Ablage 238, 323
  Datenablage 21
  Organisation 238
Ablage-, lokale Erweiterung, MIME-Typ 179
Aktivität 74, 323
  Aktivitätsgruppen 97
  Rollen zuteilen 96
Aktivitätsgruppen festlegen 97
Alias Objekte 188
Anhang 307
Applikation Symbole für Kundenprojekte 316
Applikations- und wichtige System-Objektklassen 309
Attribut 323
Attribute bearbeiten 128
Attributprofil 323
Aufsetzen der Parameter 271
Aufstellung Objektklassentypen 160
Aufzurufende URL 151
Auslösen der Meldung 112
Authentifikation mit iTAN 207
AutoUpdate 232

B

Basismodul 13
Basis-Objektklassen 324
Before-Statement 40, 80
Benachrichtigung Konfigurieren 252
Benutzer erzeugen 192
Berechtigte anzeigen 300
Berechtigungsstufe 324
Bildsymbole für die Objektklasse 168
BUTTON – Formular Schaltflächen 133

C

CHOSE / CHOOSEN – mehrere Auswahlfelder 133, 138
Clone Methoden 275
Constraint - Filter 282, 298
Customizing 11
Cut&Paste Operationen 63, 66, 86, 259

D

Datenablage 21
Datenaustausch initiieren 228
Datenbank Standards 24
Daten-Checker 129, 145
Datenimport 193, 216
Datenwächter 122, 324
Default-Statement 53
Der Prozess 62
Der Prozessstatus 65
DICOM Schnittstelle 219
Die Ablage 56, 238
Die Aktivität 74
Die Benutzervorlage 194
Die Methode 86, 99
Direkter URL Aufruf 221
Distribution 268
  Aufsetzen der Parameter 271
  Dynamische Systemrollen 270
  Parameter-Statement 48
  Unterschied Distribution – SubDistribution 274
Docbox Service erzeugen 226
Dokument 324
Dokumentarten 324
Dokumenteigenschaften 325
Dynamische Systemrollen 270
Dynamischer Prozesswechsel 267
Dyn-Statement 54

E

Eigenschaften 77, 124
Eine neue Dateiablage 239
Eine Objektklasse erzeugen 164, 281
Einen neuen Prozess erstellen 63
Einen neuen Status erzeugen 65, 103
Eingabe Validierung 122, 130, 145
Eingabeelemente 201
Eingabemasken und Listenanzeige 124
Einstellung der Speicheroption und Dateisäuberung 179
Eintrag in IPC Tabelle 108
Eintrag in Service Tabelle 111
Eintrag in Subscription Tabelle 109

E-Mail Versand (veraltet) 113
E-Mail Vorlage 114
E-Mails Abonnieren 117
E-Mails im Workflow 113
E-Mails in Überwachung 120
Ereignisse - Event Handling 102, 105
Erzeugen und Speichern 34, 59
Event-Handling 102
  Expiration-Statement 44
  Fork-Statement 44
Exchange/Outlook Anbindung 209
Expiration-Statement 44
Extension (Dokumenttyp) 325

F

Fenster 196
Festerframes 200
Finder
  Finder & Constraint-Statement 52
  Sucher 277
  Volltextsucher 281
Finder & Constraint-Statement 52
Fixer Prozessstatus 260
Fork-Statement 44
Forms Handling 229
FTP-Server 21

G

Generell 9
Generelle Arbeiten 22
Genereller Ablauf einer Aktivität 75
Geschäftsprozesse 62, 302

H

Häufig gestellte Fragen 302
Hauptfenster 196
HDR-File – Daten für Drittanwendungen 133, 137
HIDDEN – verborgene Felder 133, 137
Hinweise 119
Hinweise zu Show-Typen 128, 129, 133
HL7 Schnittstelle 217

I

Implementierung UMS 107
Importprozedur 217
Individualisierung 236
InfoPath 133, 229
Inhalt dieses Buches 9
Integration 204
Integration in HTML-Seite 221
IPC
  IPC - Interprozess Kommunikation 243
  IPC und Projekt-Methoden kombinieren 258
  Synchron/Asynchrone Aktivitäten 247
  Unified Nessaging System 108
IPC - Interprozess Kommunikation 103, 108, 111, 243
IPC und Projekt-Methoden kombinieren 258
iTAN 207

K

Katalog - Standard Symbole für Objektklassen 169
Katalog - Standard Symbole für Prozessstatus 70
KIS-Link - Integration in Klinik Informationssystem 223
Klassifikation 325
Kollektoren 297
Konfigurationsbeispiele 259
Konfigurieren der Menüleiste 150
Konfigurieren OracleDrive 213
Konfigurieren Windows WebFolder 214
Kunde 19

L

Leitdokument 325
Logging 60

M

Maske für neue Objekte 133, 140
Maskenaufbau 125
Menüleiste & Strukturanzeige 149, 194, 196, 289
Merkblatt Nomenklatur 307
Messages definieren 227
Methode 99
Module 18

N

Nomenklatur 89, 126
  Aktivität 89
  Attributprofil 126
  Datenbank Standards 24
  Oracle Bezeichner 29
  PL 55
  SQL-Statements 36
Nomenklatur Oracle Bezeichner 29
Nomenklatur PL/SQL-Funktionen 55
Nomenklatur SQL-Statements 36

O

Objekt Extension 172
Objekt Transfer 259
Objekte – Dokumente & Ordner 157

Objekte im Datenfenster anzeigen 153
Objektklasse 325
Bildsymbole f 168
Erzeugen 164
Objekt Extension 172
Objektklassen und - typen 158
Symbol Aktionen 170
Verfügbare Typen 160
Vorlagen 182
Zugriffsrechte 168
Objektklassen abgeleitet aus Objektklassentypen 158
Objektklassentyp 326
Ordner 326
Ordnerstruktur 13, 34
Organisation der Datenablage 238

P

Parameter-Statement 48, 273
Parametrisieren E-Mails in Überwachung 117, 118, 121
Passwort Einstellungen 194
PL/SQL-Funktionen 55
Pop-up Fenster definieren 201, 288, 295, 297
Prinzipieller Ablauf 106
Projekt 79, 266
Projekt Methoden 254
Projekt Methoden anwenden 254
Projektfenster 86, 197
Protokoll 326
Prozess 62, 326
Prozessstatus 65, 326
Erzeugen 65
Katalog Symbole 316
Symbole für Standard Prozesszustände 70
Prozesswechsel 262, 267

R

Reports 284, 301
Resultatfeld über Auswahlliste füllen 139, 291
Rolle 327
Rollen
Aktivität zuteilen 96
Berechtigte anzeigen 300
Rollen erzeugen 189
Sicherheitsklassifikation 249
Zugriffsrechte auf neue Objekte erteilen 168
Rollen erzeugen 189
Rollen und Benutzer 189
Rollen zuteilen 96, 302

S

Server-Checker 147
Sicherheitsklassierung 327
Sicherheitsklassifikation 249
Single Sign-On 204
Speichern und Verwenden 56
Spezielle Konfigurationsaufgaben 232
SQL-Statement Typen 36
SQL-Statements 33, 277, 282
Standard Attributprofile 127
Standard Objektklassen 327
Standard Operationen 22
Startobjekt 150
Statusbezeichnungen 66, 70
Strukturanzeige modifizieren 133, 155
Strukturelemente 327
Sucher 201, 277
Symbol Aktionen 166, 170
Synchron/Asynchrone Aktivitäten 247
Systemklassen 327

T

TAB – Registerblätter 133, 135
Tabellen in i-engine 314
Temporärverzeichnis 152
Textanpassungen 203
Tipps & Tricks 142
Typographie 10

U

Unified Messaging System - UMS 105
Unterdokument – gebunden/ungebunden 328
Unterschied Distribution – SubDistribution 274
URL Aufruf 221

V

Value-Statement 53
Variablen 40
Variablen für MS-Office Anwendungen 137, 177
Vergleich 225
Verknüpfte Tabellen 216
Verlauf 328
Versenden nach Ablauf 118
Versionsverwaltung 186
Verwalten mehrerer Vorlagentypen in einer Objektklasse 184
Verwendungsnachweis 298
VIP Durchbruch 328
Volltextsucher 166, 281
Vorgehen 11
Vorlagedateien 180
Vorlagen 328
Vorlagen für Objekte 182
Vorlagen-Objekt 182

W

Was wird im Baum angezeigt? 152
WebDav Anbindung 211
WebServices 226
Werteliste erzeugen 58, 202
Wie kann die Leistung verbessert werden? 21
Wie wird i-engine parametrisiert 12
Workflow Verknüpfung 115
Wozu sind Link/Alias Objekte? 101, 187

Z

Zeitplanen von Aufgaben 217, 237
Zugriffsrechte auf neue Objekte erteilen 168
Zwischenstatus 72